[Name of Document] Application for Patent
[Reference No.] 0306365
[Date of Filing] August 28, 2003
[Addressee] Commissioner of Japan Patent Office
[Int. Class] G03G 21/00 396
[Inventor]
[Address] c/o Ricoh Company, Ltd.
3-6, Nakamagome 1-chome, Ota-ku, Tokyo
[Name] Hiroyuki Matsushima
[Applicant]
[Id. No.] 000006747
[Address] 3-6, Nakamagome 1-chome, Ota-ku, Tokyo
[Name] Ricoh Company, Ltd.
[Representative] Masamitsu Sakurai
[Agent]
[Id. No.] 100080931
[Address] Ikebukuro Whitehouse Building #818,
20-2, Higashi-Ikebukuro 1-chome,
Toshima-ku, Tokyo
[Patent Attorney]
[Name] Hiroshi Osawa
[Priority Document]
[Application No.] 2002-276451
[Date of Filing] September 24, 2002
[Priority Document]
[Application No.] 2002-272978
[Date of Filing] September 19, 2002
[Application Fee]
[Prepayment No.] 014498
[Amount of Payment] 21,000 Yen
[List of Documents Attached]
[Name of Document] Specification 1
[Name of Document] Drawing 1

[Name of Document] Abstract 1

[Generic Authorization No.] 9809113

[Name of Document] Scope of claims

[Claim 1]

A communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which a client request corresponding to an operation request from the communication client is described, describing an operation response to the client request in the communication response, and transmitting to the communication client, the communication server being characterized by comprising:

a receiving means for receiving the client request and an operation response to a server request corresponding to the operation request transmitted to the communication client, both of which are described in the communication request, in one batch from the communication client;

a transmission means for transmitting an operation response to the client request received from the communication client and the server request, as a communication response to the communication request, in one batch to the communication client; and

a means for executing an operation related to the client request and generating an operation response to the client request as a result of the execution.

[Claim 2]

The communication server according to claim 1, characterized in that the operation request is a function call and the operation response is an execution result of the function called by the function call.

[Claim 3]

A communication server for receiving an communication request from an communication client, transmitting to the communication

client a communication response to the communication request, receiving from the communication client a communication request in which the client request corresponding to an operation request from the communication client is described, describing an operation response to the client request in the communication response, and transmitting to the communication client, the communication server being characterized by including:

a first storing means for storing the client request and an operation response to the request;

a second storing means for storing a server request corresponding to the operation request to the communication client and an operation response to the request;

a request generation means for generating the server request and storing in the second storing means;

a response generation means for reading a client request from the first storing means, executing an operation related to the client request, generating an operation response to the client request as a result of the execution, and storing in the first storing means in association with the client request which has read the operation response;

a receiving means for receiving the client request and an operation response to the server request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client;

a distribution means for storing the client request received by the receiving means in the first storing means and the operation response to the server request transmitted to the communication client, which has been received by the receiving means, in the second storing means in association with the server request transmitted to the communication client;

a collection means for reading an operation response to the client request from the first storing means and the server request

from the second storing means; and

a transmission means for transmitting the operation response read by the collection means and the server request to the communication client in one batch, as a communication response to the communication request received by the receiving means.

[Claim 4]

The communication server according to claim 3, characterized in that:

the receiving means receives an operation response and an operation request being received from the communication client as a SOAP message respectively; and

the transmission means transmits an operation response and operation request, both of which are to transmit to the communication client, as a SOAP message respectively.

[Claim 5]

The communication server according to claims 3 and 4, characterized by:

comprising a means for assigning apriority order to a client request stored in the first storing means and to a server request stored in the second storing means;

arranging that the response generation means servers as a means for reading client requests in order of a higher priority, generating an operation response to the client request, and storing in the first storing means; and

arranging that the collection means serves as a means for reading operation responses to the client request in order of a higher priority from the first storing means and server requests in order of a higher priority from the second storing means.

[Claim 6]

A communication server for receiving an HTTP request from a communication client, transmitting to the communication client an HTTP response to the HTTP request, receiving from the communication client a SOAP request of the communication client,

which has been described in an HTTP request, describing a SOAP response to the SOAP request in the HTTP response, and transmitting to the communication client, the communication server being characterized by including:

a receiving means for receiving a SOAP request from the communication client and a SOAP response to the SOAP request transmitted to the communication client, both of which have been described in one HTTP request, from the communication client;

a transmission means for describing in one HTTP a SOAP response to the SOAP request from the communication client and a SOAP request to the communication client and transmitting to the communication client as an HTTP response to the HTTP request; and

a means for executing an operation related to the SOAP request received from the communication client, and generating an execution result being described in the SOAP response to the SOAP request.

[Claim 7]

The communication server according to claim 6, characterized in that a function call is described in the SOAP request and an execution result of the function called by the function call is described in the SOAP response.

[Claim 8]

A communication server for receiving an HTTP request from a communication client, transmitting to the communication client an HTTP response to the HTTP request, receiving from the communication client a SOAP request from the communication client, which has been described in a HTTP request, describing a SOAP response to the SOAP request in the HTTP response and transmitting to the communication client, the communication server being characterized by including:

a first storing means for storing a client request corresponding to an operation request from the communication client and an operation response to the request;

a second storing means for storing a server request corresponding to an operation request to the communication client and an operation response to the request;

a request generating means for generating the server request and storing in the second storing means;

a response generating means for reading a client request from the first storing means, executing an operation related to the client request, generating an operation response to the client request as a result of the execution, and storing the operation response in the first storing means in association with the read client request;

a receiving means for receiving from the communication client a SOAP request describing the client request and a SOAP response corresponding to the SOAP response to the SOAP request transmitted to the communication client and having been described with an operation response to the server request sent to the communication client, which have been described in one HTTP request;

a distribution means for storing in the first storing means a content of the client request being described in the SOAP request received by the receiving means, and storing in the second storing means a content of the operation response being described in the SOAP response received by the receiving means in association with server request transmitted to the communication client;

a collection means for reading an operation response to the client request from the first storing means and the server request from the second storing means; and

a transmitting means for transmitting, to the communication client, a SOAP response being described with a content of the operation response read by the collection means and a SOAP request being described with a content of the operation request read by the collection means, both of which have been described in one HTTP response as an HTTP response to the one HTTP request.

[Claim 9]

The communication server according to claim 8, characterized in that:

a means for assigning a priority order to a client request stored in the first storing means and to a server request stored in the second storing means is provided;

the response generation means serves as a means for reading client requests in order of a higher priority, generating an operation response to the client request, and storing in the first storing means; and

the collection means serves as a means for reading operation responses to the client request in order of a higher priority from the first storing means and the server requests in order of a higher priority from the second storing means.

[Claim 10]

A control method of communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication request a communication request in which a client request corresponding to an operation request from the communication client is described, describing an operation response to the client response in the communication response, and transmitting to the communication client, the control method of communication server being characterized by causing the communication server to execute procedures of:

receiving the client request and an operation response to the server request corresponding to an operation request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client;

transmitting, to the communication client, an operation response to the client request received from the communication client and the server request in one batch as a communication

response to the communication request; and

executing an operation related to the client request and generating an operation response to the client request as a result of the execution.

[Claim 11]

The communication server according to claim 10, characterized in that the operation request is a function call and the operation response is an execution result of the function called by the function call.

[Claim 12]

A control method of communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which a client request corresponding to an operation request from the communication client is described, describing an operation response to the client request in the communication response, and transmitting to the communication client, the control method of communication server being characterized by causing the communication server to execute procedures of:

arranging a first storing area for storing the client request and an operation response to the request;

arranging a second storing area for storing a server request corresponding to an operation request to the communication client and an operation response to the request;

generating the server request and storing in the second storing area;

reading a client request from the server request, executing an operation related to the client request, generating an operation response to the client request as a result of the execution, and storing in the storing area in association with the client request which has read the operation response;

receiving the client request and an operation response to a server request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client;

storing in the first storing area a client request received in the receiving procedure, and storing in the second storing area an operation response to the server request transmitted to the communication client, which has been received in the receiving procedure, in association with the server request transmitted to the communication client;

reading from the first response an operation response to the client request and the server request from the second storing area; and

transmitting an operation response and an server request, both of which have been read in the collection procedure, in one batch to the communication client.

[Claim 13]

The control method of communication server according to claim 12, characterized in that:

in the receiving procedure, an operation response and an operation request to be received from the communication client are received as a SOAP message respectively; and

in the receiving procedure, an operation response and an operation request to be transmitted to the communication client are transmitted as a SOAP message respectively.

[Claim 14]

The control method of communication server according to claim 12 or 13, characterized by causing:

the communication server to execute a procedure for assigning a priority order to a client request stored in the first storing area and to a server request stored in the second storing area;

the response generation procedure to read client request in order of a higher priority, generate an operation response to the

client request, and store in the first storing area; and

the collection procedure to read from the first storing area operation responses to the client request in order of a higher priority, and read from the second storing area server requests in order of a higher priority.

[Claim 15]

A program for causing a computer to serve as a communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which a client request corresponding to an operation request is described, and describing an operation response to the client request in the communication response, and transmitting to the communication client, the program being characterized by including a program for causing the computer to serve as:

a receiving means for receiving the client request and an operation response to a server request corresponding to an operation request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client;

a transmission means for transmitting an operation response to a client request received from the communication client and the server request in one batch to the communication client, as a communication response to the communication request; and

a means for executing an operation related to the client request and generating an operation response to the client request as a result of the execution.

[Claim 16]

The program according to claim 15, characterized in that:

the operation request is a function call; and

the operation response is an execution result of the function called by the function call.

[Claim 17]

A program for causing a computer to serve as a communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which a client request corresponding to an operation request from the communication client is described, describing an operation response to the client request in the communication response, and transmitting to the communication client, the program characterized by including a program for causing the computer to serve as:

a first storing means for storing the client request and an operation response to the request;

a second storing means for storing a server request corresponding to an operation request to the communication client and an operation response to the request;

a request generation means for generating and storing the server request in the second storing means;

a response generation means for reading a client request from the first storing means, executing an operation related to the client request, generating an operation response to the client request as a result of the execution and storing in the first storing means in association with the client request which has read the operation response;

a receiving means for receiving the client request and an operation response to the server request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client;

a distribution means for storing in the first storing means a client request received by the receiving means and storing in the second storing means an operation response to the server request

transmitted to the communication client, which has been received by the receiving means, in association with the server request transmitted to the communication client;

a collection means for reading an operation response to the client request from the first storing means and the sever request from the second storing means; and

a transmission means for transmitting the operation response and the server request, both of which have been read by the collection means, in one batch to the communication client, as a communication response to a communication request received by the receiving means.

[Claim 18]

The program according to claim 17, characterized in that:

the receiving means serves as a function of receiving an operation response and an operation request being received from the communication client as a SOAP message respectively;

the transmission means serves as a function for transmitting an operation response and an operation request being transmitted to the communication client as a SOAP message respectively.

[Claim 19]

The program according to claim 17 or 18, characterized by:

further including a program for serving as a means for assigning a priority order to a client request stored in the first storing means and to a server request stored in the second storing means;

serving the response generation means as a function for reading client requests in order of a higher priority, generating and storing an operation response to the client request in the first storing means; and

serving the collection means as a function for reading operation responses to the client request in order of a higher priority from the first storing means and reading server requests in order of a higher priority order from the second storing means.

[Claim 20]

A computer readable recording medium in which the program according to any one of claims 15 to 19 is recorded.

[Name of document] Specifications

[Name of the Invention] Communication Server, Control Method of Communication Server, Program, and Recording Medium

[Technical Field]

[0001]

The present invention relates to a communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which an operation request from the communication client is described, and transmitting to the communication client a communication response in which an operation response to the operation request is described, a control method of the communication serer, a program for causing a computer to serve as the communication server, and a computer readable recording medium in which such a program is recorded.

[Background Art]

[0002]

Traditionally, in a communication system being connected with a communication apparatus through a network, communication and request are performed to a target communication apparatus by exchanging a message between each other. Then, in such a system, an operation is performed by causing a device to transmit a command as an operation request to another device and an execution result of the operation is returned by the transmitted device as an operation response.

Also, a communication system is known, in which a part of a communication apparatus making up the communication system is configured as a communication client and the other part of the communication apparatus is configured as a communication server and

communication between the communication client and the communication server is performed by a protocol in which a communication request is transmitted always from the communication client to the communication server and a communication response from the communication server to the communication client being the sender of the request.

Therein, it is performed that an operation request from a communication client to a communication server is described in a communication request and transmitted, and that an operation response to the operation request is described in a communication response and returned from the communication server to the communication client.

[0003]

Also, as a technology in which an operation is performed by transmitting an operation request from a communication server to a communication client, the following is known.

For example, it is described in patent document 1 that a remote processor transmits a message for specifying a command to be executed to a local processor and receives a response to the command.

In the document, a technology is also disclosed that when a local processor is arranged inside of a firewall, a command may be transmitted from the outside of the firewall to the inside by transmitting a communication request from the local processor to a remote process at the outside of the firewall and transmitting a command by the remote processor to the local processor as a response to the communication request.

In this case, the local processor corresponds to a communication client and the remote processor corresponds to a communication server.

[Patent document 1] Japanese Patent Laid Open Publication No. 2001-273211

[0004]

Technologies related to such operation requests may be applied to a system for remote controlling an operation of a device connected with a communication apparatus. In the patent document 2, an example

is described that such technology is applied to a remote operation device system for causing a remote target operation device having an operation function of a blind and a light to operate a blind and a light by transmitting a command from a remote operation device having a function for accepting an operation from a user. However, transmitting a response to the command is not mentioned in the document.

[Patent document 2] Japanese Patent Laid Open Publication No. 2002-135858

[Disclosure of the Invention]

[Problems to be Solved by the Invention]

[0005]

When messages are exchanged among a plurality of communication apparatuses, a communication apparatus to send commands are not necessarily limited to one. Alternatively, a plurality of communication apparatuses may be arranged to send commands to each other, in which case the communication apparatus receiving a command is requested to send back an execution result to the sender of the command. In such an arrangement, information sent from one communication apparatus to a counterpart communication apparatus may be a command to the counterpart communication apparatus or an execution result of a command received from the counterpart communication apparatus.

[0006]

In the conventional art, these commands and execution results are sent separately. In such a method, separate connections are respectively established for the time a command is to be sent and for the time an execution result of the received command is to be sent. In turn, the communication overhead is increased and thereby a problem in communication efficiency arises.

Presently, there are still many environments implementing the dial-up connection for establishing connection via a network. The above-described problem is particularly troublesome in such

environments. Namely, in such environments, it may take several tens of seconds to establish a connection, and a fee is charged each time a connection is established. Therefore, an increase in the number of connections established results in a significant rise in costs.

The present invention is for solving the above-described problems and its object is to improve communication efficiency for the transmission/reception of an operation request and an operation response to the operation request among a plurality of communication apparatuses.

[Means for solving the problems]

[0007]

For achieving the above objectives, in a communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which a client request corresponding to an operation request from the communication client is described, describing an operation response to the client request in the communication response, and transmitting to the communication client, the communication server of the present invention includes a receiving means for receiving the client request and an operation response to a server request corresponding to the operation request transmitted to the communication client, both of which are described in the communication request, in one batch from the communication client, a transmission means for transmitting an operation response to the client request received from the communication client and the server request, as a communication response to the communication request, in one batch to the communication client, and a means for executing an operation related to the client request and generating an operation response to the client request as a result of the execution.

In the communication server according to the above, the

operation request may be a function call and the operation response may be an execution result of the function called by the function call.

[0008]

In a communication server for receiving an communication request from an communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which the client request corresponding to an operation request from the communication client is described, describing an operation response to the client request in the communication response, and transmitting to the communication client, the invention provides a communication server including a first storing means for storing the client request and an operation response to the request, a second storing means for storing a server request corresponding to the operation request to the communication client and an operation response to the request, a request generation means for generating the server request and storing in the second storing means, a response generation means for reading a client request from the first storing means, executing an operation related to the client request, generating an operation response to the client request as a result of the execution, and storing in the first storing means in association with the client request which has read the operation response, a receiving means for receiving the client request and an operation response to the server request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client, a distribution means for storing the client request received by the receiving means in the first storing means and the operation response to the server request transmitted to the communication client, which has been received by the receiving means, in the second storing means in association with the server

request transmitted to the communication client, a collection means for reading an operation response to the client request from the first storing means and the server request from the second storing means, and a transmission means for transmitting the operation response read by the collection means and the server request to the communication client in one batch, as a communication response to the communication request received by the receiving means.

[0009]

In the communication server according to the above, the receiving means may receive an operation response and an operation request being received from the communication client as a SOAP message respectively, and the transmission means may transmit an operation response and operation request, both of which are to transmit to the communication client, as a SOAP message respectively.

[0010]

Further, a means for assigning apriority order to a client request stored in the first storing means and to a server request stored in the second storing means may be provided, the response generation means may server as a means for reading client requests in order of a higher priority, generating an operation response to the client request, and storing in the first storing means, and the collection means may serve as a means for reading operation responses to the client request in order of a higher priority from the first storing means and server requests in order of a higher priority from the second storing means.

[0011]

In a communication server for receiving an HTTP request from a communication client, transmitting to the communication client an HTTP response to the HTTP request, receiving from the communication client a SOAP request of the communication client, which has been described in an HTTP request, describing a SOAP

response to the SOAP request in the HTTP response, and transmitting to the communication client, the present invention provides a communication server including a receiving means for receiving a SOAP request from the communication client and a SOAP response to the SOAP request transmitted to the communication client, both of which have been described in one HTTP request, from the communication client, a transmission means for describing in one HTTP a SOAP response to the SOAP request from the communication client and a SOAP request to the communication client and transmitting to the communication client as an HTTP response to the HTTP request, and a means for executing an operation related to the SOAP request received from the communication client, and generating an execution result being described in the SOAP response to the SOAP request.

In the communication server according to the above, a function call may be described in the SOAP request and an execution result of the function called by the function call may be described in the SOAP response.

[0012]

In a communication server for receiving an HTTP request from a communication client, transmitting to the communication client an HTTP response to the HTTP request, receiving from the communication client a SOAP request from the communication client, which has been described in a HTTP request, describing a SOAP response to the SOAP request in the HTTP response and transmitting to the communication client, the present invention also provides a communication server including a first storing means for storing a client request corresponding to an operation request from the communication client and an operation response to the request, a second storing means for storing a server request corresponding to an operation request to the communication client and an operation response to the request, a request generating means for generating the server request and storing in the second storing

means, a response generating means for reading a client request from the first storing means, executing an operation related to the client request, generating an operation response to the client request as a result of the execution, and storing the operation response in the first storing means in association with the read client request, a receiving means for receiving from the communication client a SOAP request describing the client request and a SOAP response corresponding to the SOAP response to the SOAP request transmitted to the communication client and having been described with an operation response to the server request sent to the communication client, which have been described in one HTTP request, a distribution means for storing in the first storing means a content of the client request being described in the SOAP request received by the receiving means, and storing in the second storing means a content of the operation response being described in the SOAP response received by the receiving means in association with server request transmitted to the communication client, a collection means for reading an operation response to the client request from the first storing means and the server request from the second storing means, and a transmitting means for transmitting, to the communication client, a SOAP response being described with a content of the operation response read by the collection means and a SOAP request being described with a content of the operation request read by the collection means, both of which have been described in one HTTP response as an HTTP response to the one HTTP request.

[0013]

In the communication server according to the above, a means for assigning a priority order to a client request stored in the first storing means and to a server request stored in the second storing means may be provided, the response generation means may serve as a means for reading client requests in order of a higher priority, generating an operation response to the client request,

and storing in the first storing means, and the collection means may serve as a means for reading operation responses to the client request in order of a higher priority from the first storing means and the server requests in order of a higher priority from the second storing means.

[0014]

Further, in a control method of communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication request a communication request in which a client request corresponding to an operation request from the communication client is described, describing an operation response to the client response in the communication response, and transmitting to the communication client, the control method of communication server of the present invention causes the communication server to execute procedures of receiving the client request and an operation response to the server request corresponding to an operation request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client, transmitting, to the communication client, an operation response to the client request received from the communication client and the server request in one batch as a communication response to the communication request, and executing an operation related to the client request and generating an operation response to the client request as a result of the execution.

In the communication server according to the above, the operation request may be a function call and the operation response may be an execution result of the function called by the function call.

[0015]

Also, in a control method of communication server for

receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which a client request corresponding to an operation request from the communication client is described, describing an operation response to the client request in the communication response, and transmitting to the communication client, the present invention provides a control method of communication server for causing the communication server to execute procedures of arranging a first storing area for storing the client request and an operation response to the request, arranging a second storing area for storing a server request corresponding to an operation request to the communication client and an operation response to the request, generating the server request and storing in the second storing area, reading a client request from the server request, executing an operation related to the client request, generating an operation response to the client request as a result of the execution, and storing in the storing area in association with the client request which has read the operation response, receiving the client request and an operation response to a server request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client, storing in the first storing area a client request received in the receiving procedure, and storing in the second storing area an operation response to the server request transmitted to the communication client, which has been received in the receiving procedure, in association with the server request transmitted to the communication client, reading from the first response an operation response to the client request and the server request from the second storing area, and transmitting an operation response and an server request, both of which have been read in the collection procedure, in one batch

to the communication client.

[0016]

In the control method of communication server according to the above, in the receiving procedure, an operation response and an operation request to be received from the communication client may be received as a SOAP message respectively, and in the receiving procedure, an operation response and an operation request to be transmitted to the communication client may be transmitted as a SOAP message respectively.

[0017]

Further, the communication server may be caused to execute a procedure for assigning a priority order to a client request stored in the first storing area and to a server request stored in the second storing area, the response generation procedure may be caused to read client request in order of a higher priority, generate an operation response to the client request, and store in the first storing area, and the collection procedure may be caused to read from the first storing area operation responses to the client request in order of a higher priority, and read from the second storing area server requests in order of a higher priority.

[0018]

In a program for causing a computer to serve as a communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which a client request corresponding to an operation request is described, and describing an operation response to the client request in the communication response, and transmitting to the communication client, the program of the present invention includes a program for causing the computer to serve as a receiving means for receiving the client request and an operation response to a

server request corresponding to an operation request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client, a transmission means for transmitting an operation response to a client request received from the communication client and the server request in one batch to the communication client, as a communication response to the communication request, and a means for executing an operation related to the client request and generating an operation response to the client request as a result of the execution.

In the program according to the above, the operation request may be a function call, and the operation response may be an execution result of the function called by the function call.

[0019]

Further, in a program for causing a computer to serve as a communication server for receiving a communication request from a communication client, transmitting to the communication client a communication response to the communication request, receiving from the communication client a communication request in which a client request corresponding to an operation request from the communication client is described, describing an operation response to the client request in the communication response, and transmitting to the communication client, the present invention provides a program including a program for causing the computer to serve as a first storing means for storing the client request and an operation response to the request, a second storing means for storing a server request corresponding to an operation request to the communication client and an operation response to the request, a request generation means for generating and storing the server request in the second storing means, a response generation means for reading a client request from the first storing means, executing an operation related to the client request, generating an operation response to the client request

as a result of the execution and storing in the first storing means in association with the client request which has read the operation response, a receiving means for receiving the client request and an operation response to the server request transmitted to the communication client, both of which have been described in the communication request, in one batch from the communication client, a distribution means for storing in the first storing means a client request received by the receiving means and storing in the second storing means an operation response to the server request transmitted to the communication client, which has been received by the receiving means, in association with the server request transmitted to the communication client, a collection means for reading an operation response to the client request from the first storing means and the sever request from the second storing means, and a transmission means for transmitting the operation response and the server request, both of which have been read by the collection means, in one batch to the communication client, as a communication response to a communication request received by the receiving means.

[0020]

In the program according to the above, the receiving means may serve as a function of receiving an operation response and an operation request being received from the communication client as a SOAP message respectively, the transmission means may serve as a function for transmitting an operation response and an operation request being transmitted to the communication client as a SOAP message respectively.

[0021]

Further, a program for serving as a means for assigning a priority order to a client request stored in the first storing means and to a server request stored in the second storing means may be included, the response generation means may serve as a

function for reading client requests in order of a higher priority, generating and storing an operation response to the client request in the first storing means, and the collection means may serve as a function for reading operation responses to the client request in order of a higher priority from the first storing means and reading server requests in order of a higher priority order from the second storing means.

[0022]

The recording medium of the present invention is a computer readable recording medium in which the program according to any one of the above is recorded.

[Effect of the Invention]

[0023]

According to a communication server and a control method of communication server of the present invention as described in above, in a case where a plurality of communication apparatuses for receiving/transmitting communication requests and communication response to the requests receive/transmit operation requests and operation responses to the received operation requests each other, communication efficiency may be improved. Also, according to a program of the present invention, by causing a computer to serve as a communication apparatus for realizing the characteristics, the similar effect may be obtained. According to a recording medium of the present invention, by causing a computer which does not store the program to read and execute the program, the above effect may be obtained.

[Best Mode for Carrying out the Invention]

[0024]

In the following, a best embodiment of the invention is described referring to drawings.

Fig. 1 shows an exemplary configuration of a communication system of the invention configured using a communication apparatus of the invention.

The communication system is configured by connecting first communication apparatus 1 and second communication apparatus 2 being communication apparatus for the invention respectively by a network 10 as shown in Fig. 1.

Then, the first communication apparatus 1 and the second communication apparatus 2 may be configured as a various type electronic apparatus including communication function and information process function as well as a computer such as a PC including communication function. As for the network 10, the Internet, a LAN (local area network), or other various communication routes that enable network communication may be used regardless of whether it is wired or wireless.

[0025]

The first communication apparatus 1 and the second communication apparatus 2 are mounted with various application programs for controlling and managing each other. Then, each node is arranged to send an 'operation request', which corresponds to a request for an execution of a method (process) of an application program implemented in each node, and acquire an 'operation response', which corresponds to a processing result of the requested process, using RPC (remote procedure call). In other words, the first communication apparatus 1 is capable of generating a request to the second communication apparatus 2 (referred to as first communication apparatus request, hereinafter), transmitting the first communication apparatus request to the second communication apparatus 2, and acquiring a response to this request. The second communication apparatus 2 is capable of generating a request to the first communication apparatus 1 (referred to as second communication apparatus request, hereinafter), transmitting the second communication apparatus request to the first communication apparatus 1, and acquiring a response to this request.

It is noted that in the present application, a method is defined as a logical function prescribing an input and output format. Thus, the operation request corresponds to a function call (Procedure Call)

for calling the function, and the operation response corresponds to the execution result of the function called by the procedure call.

[0026]

Fig.2 shows a relation between the operation request and the operation response.

Fig. 2(A) illustrates a case in which an operation request to the second communication apparatus 2 is generated at the first communication apparatus 1. In this model, the first communication apparatus 1 generates a first communication apparatus operation request and sends this to the second communication apparatus 2, and the second communication apparatus 2 receiving this request sends back an operation response to this request.

[0027]

Fig. 2(B) illustrates a case in which an operation request to the first communication apparatus 1 is generated at the second communication apparatus 2. In this model, the second communication apparatus 2 generates a second communication apparatus operation request and sends this to the first communication apparatus 1, and the first communication apparatus 1 receiving this request sends back an operation response to this request.

It is noted that in the present embodiment, SOAP (simple object access protocol) is used as a protocol for transmitting the argument and return value by an RPC, and the above-described operation request and operation response are referred to as SOAP messages.

[0028]

The present invention is characterized in that, in an arrangement where a plurality of communication apparatuses transmit/receive operation requests and operation responses to received operation requests to/from each other, as described above, for example, an operation request that is to be sent to a counterpart communication apparatus and an operation response to an operation request received from the counterpart communication apparatus are combined and collectively sent.

In the present invention, as a communication protocol for transmitting an operation request or an operation response, for example, HTTP (HyperText transfer Protocol) or SMTP (Simple Mail Transfer Protocol) may be used as an appropriate communication protocol depending on the configuration of the communication system. However, since no corresponding relation such as a communication request and a communication response is involved in an electronic email to be sent using SMTP, a communication apparatus using SMTP is not included in the embodiment of the communication client of the invention.

Thus, first, an embodiment using HTTP as the communication protocol is described, and then a case of using SMTP as the communication protocol is described as a reference example.

[0029]

[Embodiment Using HTTP: Figs. 3 to 21]

Fig. 3 shows an exemplary configuration of a communication system applying an embodiment using HTTP. As is shown in Fig. 3, this communication system includes an HTTP server 12, an HTTP client 11, and Internet 13 that interconnects the HTTP server 12 and the HTTP client 11. The HTTP client 11 is connected to the Internet 13 via a firewall 14 to ensure security. The HTTP server 12 is a communication server and corresponds to the first communication apparatus, and the HTTP client 11 is a communication client and corresponds to the second communication apparatus.

[0030]

In performing communication using HTTP, a node stationed within the firewall 14 may not be freely accessed from outside of the firewall, and data is sent to the node only as a communication response (HTTP response) to a communication request (HTTP request) sent from this node. In this communication system, the HTTP client 11 corresponds to the node stationed within the firewall 14 and the HTTP server 12 corresponds to the node stationed outside the firewall 14. Therefore, in implementations other than the communication between the above

nodes, these nodes are not limited to functioning as a server or a client.

[0031]

Also, as same as in the first communication apparatus 1 and the second communication apparatus 2 shown in Fig. 1, the HTTP server 12 and the HTTP client 11 include application programs to control and manage each other. Further, using RPC (remote procedure call), the HTTP server 12 and the HTTP client 11 are capable of sending to each other an 'operation request' corresponding to a call to perform a method (process) of an application program implemented at the other side, and acquiring an 'operation response' corresponding to the processing result of this 'operation request'.

[0032]

Fig. 4 illustrates a relation between the operation request and the operation response.

Fig. 4(A) illustrates a case in which an operation request to the HTTP server 12 is generated at the HTTP client 11. In this model, the HTTP client 11 generates a client operation request (corresponds to a client request. Also, hereinafter, referred to as 'client command') and sends this request to the HTTP server 12, and the HTTP server 12 receiving this request sends back an operation response to this command (also, hereinafter, referred to as 'command response' or simply 'response').

[0033]

Fig. 4(B) illustrates a case in which an operation request to the HTTP client 11 is generated at the HTTP server 12. In this model, the HTTP server 12 generates a server operation request (corresponding to a server request. Also, hereinafter, referred to as 'server command') and sends this request to the HTTP client 11, and the HTTP client 11 receiving this command sends back an operation response to this command.

As is described above, in the RPC level, the operation request and the operation response are equally handled between the HTTP client

11 and the HTTP server 12. However, in the communication level, both are not equally handled.

[0034]

Fig. 5 illustrates an exemplary communication sequence in the communication system according to the present embodiment.

As is shown in the drawing, in this communication system, the HTTP client 11 sends an HTTP request to the HTTP server 12 as a communication request, and the HTTP server 12 sends back an HTTP response to the HTTP client 11 as a communication response to the communication request. For example, the HTTP client 11 may send an HTTP request X and the HTTP server 12 may respond by sending back an HTTP response X, or similarly, the HTTP client 11 may send an HTTP request Y and the HTTP server 12 may respond by sending back an HTTP response Y.

[0035]

In this embodiment, an HTTP request of a communication sequence may include and sent, a client command corresponding to an operation request to be sent from the HTTP client 11 to the HTTP server 12 and a response to a server command that has been sent from the HTTP server 12 to the HTTP client 11 (command response). Similarly, an HTTP response may include and sent, a server command corresponding to an operation request to be sent from the HTTP server 12 to the HTTP client 11 and a response to a client command that has been sent from the HTTP client 11 to the HTTP server 12 (command response).

[0036]

Therefore, for example, a client command A may be described in the HTTP request X and transferred, and a corresponding command response may be described in the HTTP response X corresponding to the HTTP request X and transferred. However, a server command C is described in the HTTP response X corresponding to the HTTP request X and transferred, and a corresponding command response to the server command C is described in an HTTP request Y and transferred.

[0037]

Also, in the case of Fig. 4(A), the HTTP client 11 is able to establish

a connection with the HTTP server 12 immediately after a client command is generated to include the generated command in an HTTP request and transmit this to the HTTP server 12. In the case of Fig. 4(B), the firewall 14 implemented at the HTTP client 11 side blocks an HTTP request from the HTTP server 12 side and, therefore, the HTTP server 12 is unable to establish a connection with the HTTP client 11 to transmit a generated server command right away.

[0038]

It is also noted that any number (including 0) of client commands and responses to server commands may be described in one HTTP request, and any number (including 0) of server commands and responses to client commands may be described in one HTTP response. The contents described in one HTTP request or HTTP response are logically transferred as one bundle.

In this way, the number of connections required in transmitting information may be reduced so that communication overhead is reduced and communication efficiency is enhanced.

[0039]

Fig. 6 illustrates another exemplary communication sequence in the communication system according to the present embodiment.

In Fig. 5, a very simple sequence example has been given in order to simplify the explanation. In Fig. 6, an example is given in which the number of commands and command responses described in each HTTP request or HTTP response is varied.

When a command is received, a response to the command may not be sent back at the next transmission opportunity. For example, as a client command B shown in Fig. 6, the corresponding command response to the client command B may not be described in the HTTP response X' of the HTTP request X' and send back. Instead, the command response may be described in a subsequent HTTP response Y' and send back.

Of course, the above may apply to the server command as well. Thus, a command response to a server command described in an HTTP response may not be described in the subsequent HTTP request. The response

may be described in any further HPPT request that is transmitted after the receipt of this HTTP response and transferred.

[0040]

It is noted that each command and command response is generated independently and implemented for process, thus, when one or more commands and command responses are to be processed together to be transferred in one bundle as described above, these commands and command responses are necessary to be combined before the transmission and to be separated after the transmission. In the following, a hardware configuration of the HTTP client 11 and the HTTP server 12, a functional configuration for realizing the above described processes, and procedural steps for realizing the processes are described.

[0041]

Fig. 7 shows an exemplary hardware configuration of the HTTP client 11 and HTTP server 12.

As is shown in the drawing, each HTTP client 11 and HTTP server 12 includes a CPU 31, a ROM 32, a RAM 33, an SD (Secure Digital) card 34, a network interface card (NIC) 35, wherein a system bus 36 interconnects these elements.

[0042]

Further, these components are described. The CPU 31 is a control means for realizing overall control of the HTTP client 11 or HTTP server 12 entirely using a control program stored in the ROM 32. The ROM 32 is a read only memory storing various kinds of fixed data including the control program used by the CPU 31.

The RAM 33 is a temporary storage memory used as a working memory for the CPU 31 to perform data processing. The SD card 34 is a nonvolatile memory that is able to retain its stored contents even when the power of the apparatus is turned off. The NIC 35 is a communication means for transmitting/receiving information to/from a communication counterpart via a network such as the Internet 13.

[0043]

Fig. 8 is a block diagram illustrating a functional configuration

of the HTTP client 11 for realizing processes pertaining to commands and command responses.

Of the functions shown in Fig. 8, a client command pool 41 and a server command pool 42 are arranged to any rewritable storage means. For example, these command pools may be arranged to the SD card 34, or may be arranged to the RAM 33 and to an HDD (hard disk drive) not shown. A client command generating means 43, a server command execution result generating means 44, a transmission message collection means 45, and a received message distributing means 48 are realized by the CPU 31. Also, HTTP request transmission means 46 and HTTP response receiving means 47 are realized by the CPU 31 and the NIC 35.

[0044]

In the following, further details of the above functions are described.

The client command pool 41 corresponds to a second storage area arranged in the HTTP client 11, and registers a client command, a response to this command, and identification information of this command in association with each other. The server command pool 42 corresponds to a first storage area arranged in the HTTP client 11, and registers a server command, a response to this command, and identification information of this command in association with each other. In these command pools, information is stored by generating a command sheet adapted into a table format for each command, wherein the command and information pertaining to the command identification information and the corresponding response are associated with each other. Each storing means being arranged with each of these pools corresponds to the second and the first storing means of the HTTP client 11 respectively.

[0045]

The client command generating means 43 corresponds to a request generating means. This client command generating means 43 has functions of generating a client command, assigning identification

information (ID) for identifying this command, attaching management information for managing this command, and registering in the client command pool 41 the above information in association with each other as a client command sheet in a table format. The part generating the client command in the client command generating means 43 corresponds to an application arranged in the HTTP client 11, for example. The client command generating means 43 may have a function of prioritizing the generated client commands for causing the HTTP server 12 to execute each command.

[0046]

Fig. 9 shows an exemplary data configuration of the client command sheet of the HTTP client 11.

As is shown in this drawing, in the client command sheet of the HTTP client 11, areas for storing data for a 'command ID', a 'method name', an 'input parameter', a 'status', a 'client command execution result notifying destination', and 'output parameter' are provided. Herein, the 'command ID', the 'method name', and the 'input parameter' correspond to the client command (and the ID attached thereto), and the 'status', and the 'client command execution result notifying destination' correspond to the management information. The 'output parameter' corresponds to the content of the command response received from the HTTP server 12.

[0047]

In the following, a specific description of each of the above items is given.

First, the 'method name' corresponds to a content of a request being made to the HTTP server 12, and indicates a type of function to be called at the HTTP server 12. The 'input parameter' corresponds to data attached to the 'method name', and indicates an argument for calling a function. The 'command ID' corresponds to identification information for identifying a client command. The 'status' corresponds to data indicating a progress of a process relating to the client command, and changes from 'not transmitted' → 'waiting for

response' → 'response received' according to the progress of the process.

[0048]

The 'client command execution result notification destination' is reference information indicating a module to be notified for causing a necessary process to be executed, when a response to a client command described in a client command sheet is received. The module being referred to usually corresponds to an application program that has generated the client command. However, this is not necessarily the case. In the 'output parameter', a content of a command response is stored at the time the command response is received. The 'output parameter' is left blank until the command response is received from the HTTP server 12.

[0049]

Referring back to Fig. 8, the server command execution result generating means 44 corresponds to a response generating means and is an application program that reads and executes a server command from the server command pool 42. This server command execution result generating means 44 has functions of generating a response to a server command, and registering in the server command pool 42 the generated response to this server command in association with the command ID of this server command. The server command received from the HTTP server 12 is arranged to be stored in the server command pool 42 as a server command sheet in a table format, in association with the ID for identifying this command and management information for managing this command. Thus, the command response generated by the server command execution result generating means 44 is also registered in the server command sheet of the corresponding server command executed.

[0050]

The server command execution result generating means 44 may be provided with a function of reading a plurality kind of server commands from the server command pool 42 and generating a response to each server command. In the case where the server commands include priority

information causing the HTTP client 11 to prioritize the execution of their respective processes, the server command execution result generating means 44 may be provided with a function of reading and executing the respective processes according to the server command having the highest priority.

Also, it is noted that the server command execution result generating means 44 may correspond to, not an application program, but a module that administers the execution of the command by calling the appropriate application program for executing the server command.

[0051]

Fig. 10 shows an exemplary data configuration of a server command sheet in the HTTP client 11.

As is shown in the drawing, the server command sheet of the HTTP client 11 includes areas for storing data corresponding to a 'command ID', a 'method name', an 'input parameter', a 'status', an 'output parameter', and a 'server command notifying destination'. The 'command ID', the 'method name', and the 'input parameter' correspond to the server command (and ID attached thereto), the 'status' and the 'server command notifying destination' correspond to the management information. The 'output parameter' corresponds to the server command execution result, which indicates the content of the command response that the HTTP client 11 sends back in response to the server command.

[0052]

In the following, the content of each data of the above is described.

First, the 'method name' corresponds to the content of a request being made to the HTTP client 11, and indicates the type of function being called in the HTTP client 11. The 'input parameter' corresponds to data attached to the 'method name', and indicates the argument for calling a function. The 'command ID' corresponds to information for identifying a server command. The 'status' corresponds to data indicating the status of a process pertaining to a server command, and changes from 'not processed' → 'process complete' → 'response

made' according to the progress of the process. The 'output parameter' stores a response generated by the server command execution result generating means 44. This item is left blank until the execution of the server command is completed and the 'status' reaches 'process complete'. The 'server command notifying destination' corresponds to reference information indicating a module for executing a server command.

[0053]

Referring back to Fig. 8, the transmitting message collection means 45 corresponds to a request collection means. The transmitting message collection means has a function for reading from the server command pool 42 a command response generated by the server command execution result generating means 44 in association with the command ID of the server command corresponding to this command response, as well as reading from the client command pool 41 a client command generated by the client command generating means 43 in association with the command ID of this command, and generating an transmitting message from the read data.

When execution priority order is assigned to the command responses and/or the client commands, the transmission message collection means 45 may be arranged to read the command responses and/or client commands according to the order of execution priority.

[0054]

Herein, a transmission message corresponds to the command response or the command and the corresponding command ID being described as a SOAP message using the XML (Extensible Markup Language), which is a structured language format. Then, the transmission message collection means 45 generates a SOAP message as a transmission message for each one of command responses or commands. In such case, the corresponding command ID of each command is described in a SOAP header and the command response or the content of the client command is described in a SOAP body. In a communication using SOAP, a message called SOAP envelope (envelope) that includes the SOAP header and the

SOAP body is described using the XML format and exchanged using a protocol such as HTTP.

The generation of a SOAP message from a command and a command response may be realized by implementing an appropriate conversion program (serializer) that is generated based on WSDL (Web Service Description Language), and serializing the data.

[0055]

The HTTP request transmission means 46 corresponds to a transmission means for generating an HTTP request containing a transmission message generated by the transmission message collection means 45, and transmitting the HTTP request to the HTTP server 12. It is noted that any number of transmission messages may be contained in an HTTP request, and also, transmission messages corresponding to command responses and transmission messages corresponding to client commands may be intermingled in the HTTP request on an arbitrary basis.

Therefore, the HTTP request transmission means 46 is adapted to include all the transmission messages generated by the transmission message collection means 45 in one HTTP request for transmission regardless to whether the transmission messages correspond to a command response or a client command. It is also possible to set a limit to the number of transmission messages included in one HTTP request.

[0056]

It is noted that once the transmission message collection means 45 undertakes reading of a client command or command response and so forth, the transmission of the HTTP request is performed even when there is no data to be read and consequently no transmission messages are generated. And, the transmission message collection means 45 periodically makes an attempt to read a command response or client command. For example, the readout operation may be performed every 60 minutes by a timer.

The above arrangement is made because the HTTP server 12 is unable to send information to the HTTP client 11 unless the HTTP client 11

makes a communication request to the HTTP server 12 as described in above. Even when there is no data to be sent from the HTTP client 11, the HTTP client 11 is arranged to periodically send a communication request to the HTTP server 12 to give the HTTP server 12 an opportunity to send information to the HTTP client 11 so that information necessary to be transmitted is prevented from remaining in the HTTP server 12 over a long period of time.

[0057]

It is noted that readout operation by the transmission message collection means 45 and the subsequent transmission of HTTP requests by the HTTP request transmission means 46 may be performed at a suitable timing aside from the periodical timing. For example, when information requiring urgent transmission is registered in any one of the pools, the client command generating means 43 or the server command execution result generating means 44 may notify this to the transmission message gathering means 45 so that the readout may be performed.

[0058]

The HTTP response receiving means 47 corresponds to a receiving means and includes a function for receiving an HTTP response from the HTTP server 12. In the HTTP response, the receiving messages including a server command and the command ID associated with this command and the receiving messages including a response to a client command and the command ID associated with this command are intermingled on an arbitrary basis.

Herein, the receiving messages correspond to the commands or responses and the corresponding command IDs as SOAP messages.

[0059]

The receiving message distributing means 48 corresponds to a distributing means and includes a function for allocating and registering the data contained in the HTTP response received by the HTTP response receiving means 47 into the server command pool 42 and client command pool 41.

The server command and the command ID associated with this command are registered in a server command sheet provided in the server command pool 42. As for the response to a client command, the command ID associated with this command is collated with the command ID of each client command sheet stored in the client command pool 41 so as to specify the corresponding client command of the command response, and this command response is registered as the 'output parameter' of the corresponding client command.

The HTTP response is resolved into the respective receiving messages, and each receiving message included in this HTTP response is extracted so that data described therein is converted into a format suitable for registration in the table. This conversion may be realized by implementing an appropriate conversion program (deserializer) that is generated based on WSDL.

[0060]

Next, Fig. 11 shows an example of an HTTP request that is to be transmitted to the HTTP server 12 by the HTTP client 11 having the above-described functions.

As is shown in Fig. 11, this HTTP request has a body portion that describes a multipart message using MIME (Multipurpose Internet Mail Extension). In each part making up the multipart message, an entity header is described and a SOAP message is embedded though detailed description in figures is omitted. In the example of Fig. 11, the HTTP body of the HTTP request is made up of independent first part, second part, third part and fourth part, divided from each other by a 'MIME_boundary'. However, the number of parts which may be included in an HTTP body is not limited to four, and any number of parts including 0 may be included.

It is also noted that the SOAP envelope embedded in the HTTP request to be transmitted includes envelopes being described with a client command or envelopes being described with a response to a server command.

[0061]

Fig. 12 shows an example of an HTTP response received by the HTTP client 11 from the HTTP server 12.

As is shown in Fig. 12, the structure of the HTTP response is identical to that of the HTTP request shown in Fig. 11 except for the HTTP header. The body portion of the HTTP response is identical to the HTTP request and describes a SOAP envelope of a multipart message in accordance with MIME. It is noted that the content of the SOAP envelope varies depending on the content of the command or command response being described.

The SOAP envelope embedded in the HTTP response to be transmitted may describe a server command or a response to a client command.

[0062]

In the following, specific examples of the parts to be described in the HTTP request or HTTP response are illustrated with reference to Figs. 13 to 16.

Fig. 13 shows an example of a part describing a client command.

In this example, the 'X-SOAP-Type' header of the entity header portion describes information indicates whether the SOAP message described in this part corresponds to a SOAP request or a SOAP response. In this case, the value 'Request' indicates that the SOAP message corresponds to a SOAP request. In other words, it is indicated that the message is a SOAP message describing a command.

Also, a 'SOAPAction' header indicates the content of a SOAP request. In this case, the content of the request is described by such URI (Uniform Resource Identifier) as 'http://www.•••'. A 'SOAP header' is not attached when the SOAP message is a SOAP response'. Therefore, depending on whether or not a 'SOAPAction' header is attached, the receiving side of a message may judge whether the SOAP message is a SOAP request or a SOAP response.

[0063]

Addresses defining namespaces are indicated as attributes of the 'Envelope' tag. In this example, the namespaces specified by such URIs as 'http://www.foo.com/header' and 'http://www.foo.com/server'

are defined, in addition to the namespaces defined as the norm in SOAP. Thus, an XML tag attached with a namespace prefix 'n' indicates that the tag belongs to the namespace being specified by the URI of 'http://www.foo.com/header', and an XML tag attached with a namespace prefix 'ns' indicates that the tag belongs to the namespace being specified by the URI of 'http://www.foo.com/server'.

[0064]

In the 'SOAP header', '12345' corresponding to the ID of this client command is described as the content of the 'requestID' XML tag. In the 'SOAP body', a 'trouble notification' tag is described as information indicating the method stored in the 'method name' of the client command sheet, and the argument and other information stored in the 'input parameter' of the client command sheet are described as elements of subordinate tags such as an 'error ID' and a 'description'. In this example, the notification content of the trouble notification is described.

[0065]

Fig. 14 shows an example of a part describing a response to a client command.

In this example, the value of the 'X-SOAP-Type' header in the entity header portion is described with 'Response' to indicate that the SOAP message described in this part corresponds to a SOAP envelope describing a command response, that is a SOAP message described with a command response.

Also, the definitions of namespaces in this example are identical to those of the previous example shown in Fig. 13. Further, in the 'SOAP header', '12345' corresponding to the ID of the client command to which the response has been generated is described as the content of the 'command ID' XML tag. In the 'SOAP body', a 'trouble notification Response' tag for indicating that the part describes a response to the 'trouble notification' command is described, and the content of the command response is described in a subordinate tag. In this example, information indicating that the trouble notification

has been properly received is described. Then, this information is stored in the item of 'output parameter' of the client command sheet.

[0066]

Fig. 15 shows an example of a part describing a server command.

As in the example of Fig. 13, the information 'SOAPAction' and the 'Request', the value of the 'X-SOAP-Type' header, described in the entity header portion of this part indicates that the SOAP envelope described in this part corresponds to a SOAP request. The information of the 'SOAPAction' header indicates the contents of the SOAP request.

[0067]

Addresses defining namespaces are indicated as attributes of the 'Envelope' tag, as in the case in Fig. 13. In this example, the namespaces specified by the such URIs as 'http://www.foo.com/header' and 'http://www.foo.com/client' are defined, in addition to the namespaces defined as the norm in SOAP.

In the 'SOAP header', '98765' corresponding to the ID of this server command is described as the content of the 'command ID' XML tag. Also, in the SOAP body, a 'temperature sensor value acquisition' tag is described as information indicating the method stored in the 'method name' of the server command sheet, and the argument or some other information stored in the 'input parameter' of the server command sheet is described as an element of a subordinate tag 'sensor ID'. In this example, the ID of the sensor from which the sensor value is to be acquired is described.

Further, it is noted that the server may send such a command to the client when it receives a trouble notification from the client, to determine the cause of the trouble.

[0068]

Fig. 16 shows an example of a part describing a response to a server command.

As in the example of Fig. 14, 'Response', as the value of the 'X-SOAP-Type' header, is described in the entity header portion of this part to indicate that the SOAP message described in this part

corresponds to a SOAP response.

Also, as in the example of Fig. 15, '98765' corresponding to the ID of the server command to which this response is generated is described in the 'SOAP header' of this part as the content of the 'command ID' XML tag. In the SOAP body, a 'temperature sensor value acquisition Response' tag indicating that the part indicates the response to the 'temperature sensor value acquisition' command is described, and the content of the server command response is described in a subordinate tag. In this example, temperature value information provided by the sensor to which the value acquisition request has been made is described.

[0069]

In the following, processes performed by the HTTP client 11 having the above-described configurations and functions are described with reference to flowcharts shown in Figs. 17 to 21. The processes illustrated in these flowcharts may be realized by the CPU 31 of the HTTP 11, which executes the appropriate control programs.

[0070]

First of all, Fig. 17 shows a flowchart illustrating a basic operation flow of a message collection and distributing process.

The CPU 31 of the HTTP client 11 starts the process illustrated by the flowchart of Fig. 17 when it is time for the transmitting message collection means 45 to undertake readout of a client command and/or command response.

According to this process flow, first, a client command collection process is performed (S11). This corresponds to a process of collecting from the client command pool 41 client commands that are to be transmitted to the HTTP server 12, and also includes a process of generating from the collected data parts containing respective SOAP envelopes.

[0071]

Next, a server command execution result collection process is performed (S12). This corresponds to a process of collecting from

the server command pool 42 command responses that are to be transmitted to the HTTP server 12, and also includes a process of generating from the gathered data parts containing respective SOAP envelopes.

Then, the parts generated in the processes of steps S11 and S12 are merged so that an HTTP request containing all these parts is generated (S13). This HTTP request is then sent to the HTTP server 12 (S14).

With regard to the processes up to this point, the CPU 31 functions as the transmitting message collection means 45 in steps S11 and S12, and as the HTTP request transmission means 46 in steps S13 and S14.

[0072]

Next, an HTTP response as a communication response to the HTTP request is received from the HTTP server 12 (S15). Then, the HTTP body of the received HTTP response is divided into parts (S16). This corresponds to a process of dividing the HTTP body into components that are separated from each other by a 'MIME_boundary', and all the parts are divided.

Then, a sequence of processes corresponding to steps S17 to S19 is successively performed for each of all the divided parts. In this process sequence, first, it is determined whether a part subjected to the process describes a server command or not (S17). Then, if it is determined that the part describes a server command, a server command registration process is performed (S18). If it is determined that the part does not describe a server command, this means that the part describes a response to a client command and, thereby, a response notification process is performed (S19).

[0073]

After performing either step S18 or S19, the process goes back to step S17, and the process sequence is repeated for a next part that is subjected to the process. Thus, the process sequence of steps S17 to S19 is performed for each of all the divided parts, and the process shown in the flowchart in FIG.17 ends when this is completed.

With regard to the processes up this point, the CPU 31 functions

as the HTTP response receiving means 47 in steps S15 and S16, and as the received message distribution means 48 in steps S17 to S19.

[0074]

In the following, the operation flow of Fig. 17 is described further using flowcharts each illustrating portions of the operation flow in more detail.

FIG.18 is a flowchart illustrating a more detailed process flow of steps S11 to S14 in Fig. 17.

[0075]

According to the process, first, the CPU 31 of the HTTP client 11 collects from the client command pool 41 contents of the 'method name' and 'input parameter' of the client command sheets indicating 'not transmitted' for the 'status' as client commands to be sent, and also collects contents of the 'command ID' as IDs of the collected client commands (S21). It is noted that when 'not transmitted' is indicated as the 'status', this means that a command generated by the client command collection means 43 has not yet been sent to the HTTP server 12. Thus, commands to be sent to the HTTP server 12 may be extracted based on the information.

[0076]

Then, processes corresponding to steps S22 to S24 are successively performed on each of all the client commands collected in the step S21. In the process, first, a client command and its corresponding command ID that are subjected to the process are converted into an XML text in which information on the client command and the command ID are included in a SOAP body and a SOAP header, respectively (S22). A SOAP envelope corresponding to a part describing the subjected client command is generated (S23). Then, the 'status' of the client command sheet describing the subjected client command is changed from 'not transmitted' to 'waiting for response' (S24). When the 'status' of the client command sheet is indicated as 'waiting for response', this means that the described command has been sent to the HTTP server 12.

[0077]

After the process sequence is completed, the CPU 31 collects the contents of the 'output parameter' of server command sheets of which the 'status' is indicated as 'process completed' as command responses to server commands that are to be transmitted to the HTTP server 12, and also collects the contents of the 'command ID' as IDs of the server commands corresponding to the collected responses (S25). When the 'status' is indicated as 'process completed', this means that the response corresponding to the server command that is generated by the server command execution result generating means 44 has not yet been sent to the HTTP server 12. Thus, the command responses to be transmitted to the HTTP server 12 may be extracted based on the information.

[0078]

Then, processes corresponding to steps S26 to S28 are successively performed on each of all the command responses collected in the step S25. The process includes converting a subjected command response and the corresponding command ID collected therewith into an XML text that describes information on the command response and the corresponding command ID in a SOAP body and a SOAP header, respectively (S26), and generating a SOAP envelope corresponding to a part describing the subjected command response (S27). The process is identical to that of steps S22 and S23 except for the difference in the items being subjected to the processes. Next, the 'status' of the server command sheet describing the subjected command response is changed to 'response made' (S28). When 'response made' is indicated as the 'status', this means that the command response has been sent to the HTTP server 12.

[0079]

After the completion of the processes described above, the CPU 31 merges the parts generated in steps S23 and S27, generates a multipart HTTP request as shown in Fig. 11, and sends this to the HTTP server 12 (S29).

It is noted that the changing of the 'status' in steps S24 and S28

may be performed after the transmission of data is actually completed. In this way, even if communication error is generated, commands and command responses to be sent may be subjected to resending, whereby reliability of the system may be improved.

With the steps of the above, the processes pertaining to the transmission of the HTTP request are completed, and the operation moves on to steps S15 and onward of Fig. 17.

[0080]

Fig. 19 is a flowchart illustrating a more detailed operation flow of Fig. 17 for the processes of steps S15 and onward. The process step following step S29 of Fig. 18 corresponds to step S31 in this drawing.

According to this process flow, first, the CPU 31 of the HTTP client 11 awaits the arrival of an HTTP response to the transmitted HTTP request, and then receives the HTTP response from the HTTP server 12 (S31). Upon receiving the HTTP response, the CPU 31 analyzes its HTTP body and divides it into parts (S32).

Then, a process sequence corresponding to steps S33 to S39 is successively performed on each of the divided parts.

[0081]

In this process sequence, first, it is determined whether a part subjected to the process corresponds to a server command (S33). As is described above, a server command and a response to a client command may be included in the HTTP response and, thus, a determination of whether a part corresponds to a server command or a response to a client command is made. The determination is made based on whether a SOAPAction header is described in the part, or the determination may be made based on the content of the X-SOAP-Type header.

[0082]

Then, when it is determined in step S33 that the part does not correspond to a server command, this means that the part describes a response to a client command. In this case, the XML text describing the part is analyzed and converted into data suitable for registration

in the client command sheet (S34). Then, a client command corresponding to the response is searched from the client command pool 41, and the command response data is registered in the item 'output parameter' of the client command sheet of the corresponding client command (S35). It is noted that a command ID identical to the information indicated as the 'command ID' in the transmitted client command is attached to the command response and, thus, the search for the client command in step S35 may be conducted using this information as a key.

[0083]

After the data registration is completed, the 'status' of the client command sheet is changed to 'response received' (S36). Then, the fact that a response has been received is notified to a destination registered in the 'client command execution result notifying destination' (S37). With this notification, an application program such as that for generating the client command may be informed that a response to the generated command has been received, and may perform the appropriate processes according to the response.

[0084]

For example, when an application program for generating a trouble notification generates a client command to send a trouble notification to the HTTP server 12, this command is sent to the HTTP server 12, and the HTTP server 12 may send back a command response indicating that the command has been properly received. Then, the HTTP client 11 receiving this command response may search the client command corresponding to this command response based on the command ID included in the received command response, and register this command response in association with the searched out client command. Then, the application program for generating the trouble notification that is registered as the command execution result notifying destination of this command is notified of the fact that a response to this command has been received. The application program for receiving this notification may refer to the client command sheet and acquire the

execution result of the generated command from the item 'output parameter' of this client command sheet.

After the processes up to step S37 are completed for the subjected part, the same processes starting from step S33 are repeated for a next part if such part exists.

[0085]

On the other hand, when it is determined in step S33 that the subjected part corresponds to a server command, the XML text describing the part is analyzed and converted into data suitable for registration in the server command sheet (S38). Then a server command sheet for this server command is created and this server command sheet containing the server command and its corresponding command ID is registered in the server command pool 42 (S39). The content of the server command is registered in the items 'method name' and 'input parameter' of the server command sheet, and the command ID described in the part is registered in the item 'command ID' of the server command sheet. In the item 'server command notifying destination', reference information to the application program with which the method registered in the 'method name' is executed, for example, is registered, this reference information being determined based on correspondence information between a method and an application program, for example, that is provided beforehand. It is noted that the initial value of the 'status' corresponds to 'not processed', and the initial value of the 'output parameter' corresponds to 'NULL'.

[0086]

After the processes up to step S39 are completed, the processes starting from step S33 are repeated for a next subjected part if such part exists.

The process illustrated in the flowchart of Fig. 19 ends when the processes from step S33 to step S39 are performed on each of all the parts.

By performing the above-described processes, the HTTP client 11 is able to send to the HTTP server 12 an operation request to be

transmitted to the HTTP server 12 together with an operation response to an operation request received from the HTTP server 12 in one bundle. Also, the HTTP client 11 is able to receive from the HTTP server 12 an operation request sent by the HTTP server 12 together with an operation response to the operation request that has been sent to the HTTP server 12 in one bundle.

[0087]

It is noted that in the present embodiment, the parts to be transmitted are generated and merged together before they are transmitted, and the parts are received as a whole after which they are divided into parts for further processing; however, the present invention is not limited to this arrangement.

As for the transmission, the HTTP header may first be transmitted, after which parts are successively transmitted each time they are generated, and notification data may be sent after the transmission of the parts is completed. Even in such arrangement, as long as the data being transmitted during the process is one logically successive HTTP request having one HTTP header, the transfer of data may be transferred in one session with one negotiation process. Thus, an effect similar to that from merging the parts together before transmitting the HTTP request may be obtained as well. Further, since the required memory capacity of the buffer for storing data to be transmitted may be reduced, a low cost communication apparatus may be arranged to handle a large quantity of data.

Also, at the receiving side, the processes for each part may be successively performed each time a part is received. As in the transmission, the required memory capacity may be reduced.

[0088]

In the following, processes pertaining to an execution of a server command are described.

Fig. 20 is a flowchart illustrating an example of such processes.

The process steps of Fig. 20 pertaining to an execution of a server command are performed after step S39 of Fig. 19. That is, these steps

may be performed after the server command is registered in the server command pool 42. Herein, the CPU 31 of the HTTP client 11 functions as the server command execution result generating means 44.

[0089]

In the process, first, an application program and so forth is called based on the information corresponding to the 'server command notifying destination' of the server command sheet describing the registered server command, and data corresponding to the 'method name' and 'input parameter' are transferred to the application program so that a process pertaining to the server command is performed (S41). Although it is not shown in this flowchart, the process pertaining to the server command is executed by the CPU 31 separately.

After this process step is completed, the execution result is registered in the item 'output parameter' of the server command sheet (S42). At the same time, the 'status' of the server command sheet is changed to 'process complete' which indicates that the process is completed (S43). Then, the operation goes back to the process flow of Fig. 19.

By means of the process steps as in above, the server command may be executed and appropriate measures may be taken so that the execution result of the server command may be in a state ready for transmission to the HTTP server 12 as the command response.

[0090]

A flowchart shown in Fig. 21 illustrates another exemplary process flow pertaining to the execution of the server command that is performed independently from the process flow of Fig. 19. Herein, the CPU 31 of the HTTP client 11 also functions as the server command execution result generating means 44.

In performing the process steps of Fig. 21, the CPU 31 initiates the process shown in the flowchart of Fig. 21 at suitable timings.

[0091]

In the process of Fig. 21, it is determined whether a server command sheet of which the 'status' is indicated as 'waiting for process'

exists or not (SX1), and if it does not exist, the process is in a waiting status until such a server command sheet is added. When such server command sheets are found, the 'status' of the server command sheet is changed to 'in process' as one of the command sheets is subjected for processing (SX2).

After the process in above, the processes corresponding to the steps S41 to S43 as similar to the processes in Fig. 20 are performed so that the server command subjected for the process and described in the server command sheet is executed. After the completion thereof, the process is repeated returning to the step SX1.

[0092]

It is noted that the above processes may be performed simultaneously using a plurality of threads (e.g., four threads). Herein, since the 'status' of a server command sheet that is once subjected to the process is no longer indicated as 'waiting for process', the server command sheet is prevented from being subjected to a process more than once even if a process is started simultaneously in a plurality of threads.

[0093]

By performing the above-described processes, each server command may be executed at an arbitrary timing and, thereby, even when the execution of one command takes a long time, its subsequent processes may not necessarily be affected by this delay. Further, the execution results are successively adjusted for transmission to the HTTP server 12 as command responses according to the order in which the execution of each command is completed.

In the above, the description of the processes pertaining to the transfer of commands and command responses performed in the HTTP client 11 has been completed.

[0094]

A functional configuration of the HTTP server 12 is described. Herein, a hardware configuration of the HTTP server 12 may be identical to that of the HTTP client 11 as described using Fig. 7. In the flowing

descriptions of the HTTP server 12, numerical references for hardware components are identical to those for the HTTP client 11.

Fig. 22 is a block diagram illustrating an exemplary functional configuration of the HTTP server 12 for performing processes pertaining to the commands and command responses.

Of the functions shown in Fig. 22, a server command pool 141 and a client command pool 142 are provided in any suitable rewritable storage means of the HTTP server 12. Server command generation means 143, client command execution result generation means 144, transmission message collection means 145, and receiving message distributing means 148 are realized by the CPU 31. Also, HTTP response transmission means 146 and HTTP request receiving means 147 are realized by the CPU 31 and the NIC 35.

[0095]

In the following, further details of the above functions are described.

First, the server command pool 141 corresponds to a second storage area arranged in the HTTP server 12, and is a pool for registering a server command, a response to this command, and identification information of this command in association with each other. Also, the client command pool 142 corresponds to a first storage area arranged in the HTTP server 12, and is a pool for registering a client command, a response to this command, and identification information of this command in association with each other.

[0096]

The server command generating means 143 corresponds to request generating means for generating a server command, assigning identification information (ID) for identifying this command, attaching management information for managing this command, and registering in the server command pool 141 the above data in association with each other as a client command sheet in a table format. Further, generating the client command in the client command generating means 43 may be realized by an application program

implemented in the HTTP server 12, for example. Also, the server command generating means 143 may have a function of prioritizing the execution of the generated server commands in administering the HTTP client 11 to execute the commands.

[0097]

Fig. 23 shows an exemplary data configuration of the server command sheet of the HTTP server 12.

As is shown in this drawing, the server command sheet of the HTTP server 12 has areas for storing data that are largely identical to those of the client command sheet of the HTTP client 11 shown in Fig. 9. The difference between these two command sheets lies in the fact that the type of command described in the server command sheet of the HTTP server 12 is a server command rather than a client command and the HTTP client 11 is designated as the destination of this command and the sender of its corresponding command response. Further, as for the 'server command execution result notifying destination' of the server command sheet of the HTTP server 12, although the title of this item is different since the command sheet describes a server command rather than a client command, the actual contents that may be registered as data for this item are identical to those for the 'client command execution result notifying destination' of Fig. 9.

[0098]

Referring back to Fig. 22, the client command execution result generating means 144 corresponding to response generating means is an application program for reading a client command from the client command pool 142. This client command execution result generating means 144 has functions of generating a response to a client command, and registering in the client command pool 142 the generated response to this client command in association with the command ID of this command. Herein, it is noted that the client command received from the HTTP client 11 is arranged to be registered in association with the ID for identifying this command and management information for managing this command in the client command pool 142 as a client command

sheet in a table format. Thus, the command response generated by the client command execution result generating means 144 is also registered in the client command sheet of the corresponding client command.

[0099]

The client command execution result generating means 144 may be provided with a function of reading a plurality of client commands from the client command pool 142 and generating a response for each of the readout client commands. Further, in the case where the client commands include priority information for the HTTP server 12 to prioritize the execution of their respective processes, the client command execution result generating means 144 may be provided with a function of reading and executing the respective processes according to the order of priority starting with the client command with the highest priority.

It is noted that the client command execution result generating means 144 does not necessarily have to be an application program itself, and may instead be a module that administers the execution of a command by calling a suitable application program for executing the server command, for example.

[0100]

Fig. 24 shows an exemplary data configuration of a client command sheet in the HTTP server 12.

As is shown in this drawing, the client command sheet of the HTTP server 12 has areas for storing data that are substantially identical to those of the server command sheet of the HTTP client 11 shown in FIG.10. The difference between these two command sheets is that the type of command described in this client command sheet of the HTTP server 12 is a client command and the HTTP client 11 is designated as the sender of this command and the destination of its corresponding command response. Further, as for the 'client command notifying destination', although the title of this item is different since the command described in the command sheet is a server command, the actual

contents are identical to those for the 'server command notifying destination' of Fig. 10.

[0101]

Referring back to Fig. 22, the transmitting message collection means 145 corresponds to a collection means. The transmitting message collection means 145 has a function for reading from the client command pool 142 a command response generated by the client command execution result generating means 144 that is registered in association with the command ID of the client command corresponding to this command response, as well as reading from the server command pool 141 a server command generated by the server command generating means 143 that is registered in association with the command ID of this command, and generating an transmitting message from the read data.

When execution priority information is assigned to the command responses and/or the client commands, the transmission message collection means 145 may be arranged to read the command responses and/or client commands according to the order of execution priority.

Herein, the format of the transmission message is identical in the HTTP client 11.

[0102]

The HTTP response transmission means 146 corresponds to transmitting means for generating an HTTP request containing the transmission message generated by the transmission message collection means 145 as a communication response to the HTTP request received from the HTTP client 11, and transmitting this HTTP response to the HTTP client 11. Herein, it is noted that any suitable number of transmission messages may be contained in an HTTP response, and also, transmission messages corresponding to command responses and transmission messages corresponding to client commands may be intermingled in the HTTP response on an arbitrary basis.

Thus, the HTTP response transmitting means 146 is adapted to include the transmission messages generated by the transmission message collection means 145 in one HTTP request for transmission to

the HTTP client 11 regardless whether the transmission messages contained therein correspond to a command response or a client command. However, it is also possible to set a limit to the number of transmission messages included in one HTTP request.

[0103]

It is noted that once the transmission message collection means 145 undertakes reading of a server command or command response, the transmission of the HTTP response is performed even when there is no data to be read and consequently no transmission messages are generated. That is, the transmission message collection means 145 undertakes readout of a command response or server command upon receiving an HTTP request from the HTTP client 11.

The above arrangement is made because the HTTP server 12 is unable to send information to the HTTP client 11 unless the HTTP client 11 makes a communication request to the HTTP server 12 as described in above.

[0104]

The HTTP request receiving means 147 corresponds to receiving means for acquiring an HTTP request from the HTTP client 11. In the HTTP request, receiving messages including a client command and the command ID in association with a client command and receiving message including a response to a server command and the command ID in association with the command may be included on any arbitrary basis.

Herein, the receiving message describes the command or response and the corresponding command ID as a SOAP message.

[0105]

The receiving message distributing means 148 corresponds to distributing means for allocating and registering the data contained in the HTTP request received by the HTTP request receiving means 147 into the server command pool 141 and the client command pool 142, respectively.

Specifically, the client command and the command ID associated with this command are registered in a client command sheet provided in the

client command pool 142. As for the response to a server command, the command ID associated with this command is collated with the command ID registered in each server command sheet stored in the server command pool 141 so as to determine the corresponding server command of the command response, and this command response is registered as the 'output parameter' of the corresponding server command.

[0106]

The HTTP request is divided into respective receiving messages, and each receiving message included is extracted so that data described therein is converted into a format suitable for registration in the table. This conversion may be realized by executing an appropriate conversion program (deserializer) that is generated based on WSDL.

The HTTP request received by the HTTP server 12 having the above-described functions corresponds to a communication request sent from the HTTP client 11. Thus, for example, this HTTP request may correspond to the HTTP request of Fig. 11 used in describing the functions of the HTTP client 11. Similarly, the HTTP response sent by the HTTP server 12 corresponds to the communication response sent to the HTTP client 11 to be received by this HTTP client 11. Thus, for example, this HTTP response may correspond to that described using Fig. 12. Further, the contents of the parts included in the HTTP request and HTTP response may be identical to those described in reference to Figs. 13 to 16.

[0107]

In the following, processes performed by the HTTP server 12 having the above-described configurations and functions are described with reference to flowcharts shown in Figs. 25 to 29. Herein, it is noted that the processes illustrated in these flowcharts are realized by the CPU 31 of the HTTP server 12 that executes the appropriate control programs.

[0108]

First of all, Fig. 25 shows a flowchart illustrating a basic operation flow of a message collection and distributing process.

The CPU 31 of the HTTP server 12 starts the process illustrated by the flowchart of Fig. 25 when an HTTP request is sent from the HTTP client 11.

According to this process flow, first the HTTP request is received (S111). Then, the HTTP body of the received HTTP request is divided into parts (S112). The HTTP body is divided into components that are separated from each other by a ''MIME_boundary', and the division is made to all the parts.

[0109]

Then, a sequence of processes corresponding to steps S11 to S115 is successively performed for each of the divided parts. In the process sequence, first, it is determined whether a part subjected to the process describes a client command (S113). Then, if it is determined that the part describes a client command, a client command registration process is performed (S114). On the other hand, if it is determined that the part does not describe a client command, this means that the part describes a response to a server command and, thereby, a response notification process is performed (S115).

[0110]

After performing either step S114 or S115, the process goes back to step S113, and the process sequence is repeated for a next part that is subjected to the process. Then, after the process sequence of steps S113 to S115 is performed for each of the divided parts, the process moves on to the next step S116.

With regard to the processes up this point, the CPU 31 functions as the HTTP request receiving means 147 in steps S111 and S112, and as the received message distribution means 148 in steps S113 to S115.

[0111]

Next, the CPU 31 performs a server command collection process (S116). This corresponds to a process of collecting from the server command pool 142 server commands that are to be transmitted to the HTTP client 11, and also includes a process of generating from the collected data parts containing respective SOAP envelopes.

Next, a client command execution result collection process is performed (S117). This corresponds to a process of collecting from the client command pool 142 command responses that are to be transmitted to the HTTP client, and also includes a process of generating from the collected data parts containing respective SOAP envelopes.

[0112]

Then, the parts generated in the process steps S116 and S117 are merged together so that an HTTP response containing these parts is generated (S118). This HTTP response is then sent to the HTTP client 11 as a communication response to the HTTP request received in step S111 (S119), after which this message collecting and distributing process is completed.

With regard to the processes up to this point, the CPU 31 functions as the outgoing message collection means 145 in steps S116 and S117, and as the HTTP response transmitting means 146 in steps S118 and S119.

[0113]

In the following, the operation flow of Fig. 25 is described further using flowcharts each illustrating portions of the operation flow in greater detail.

Fig. 26 is a flowchart illustrating a more detailed operation flow of Fig. 25 for the process steps S111 to S115.

According to this process flow, first, the CPU 31 of the HTTP server 12 receives the HTTP request transmitted from the HTTP client 11 (S121). Upon receiving the HTTP request, the CPU31 analyzes its HTTP body and divides it into parts (S122).

Then, a process sequence corresponding to steps S123 to S129 is successively performed on each of the divided parts.

[0114]

In this process sequence, first, it is determined whether a part subjected to the process corresponds to a client command (S123). As is described above, a client command and a response to a server command may be included in the HTTP request and, thus, a determination of

whether a part corresponds to a client command or a response to a server command is made. This determination is made based on whether a 'SOAPAction' header is described in the part, or the determination may be made based on the content of the 'X-SOAP-Type' header.

[0115]

Then, when it is determined in step S123 that the part does not correspond to a client command, this means that the part describes a response to a server command. In this case, the XML text describing the part is analyzed and converted into data suitable for registration in the server command sheet (S124). Then, a server command corresponding to this response is searched from the server command pool 141, and the command response data is registered in the item 'output parameter' of the server command sheet of the corresponding server command (S125). It is noted that a command ID identical to the information indicated as the 'command ID' in the transmitted server command is attached to the command response and, thus, the search for the corresponding server command in step S125 may be conducted using this information as a key.

[0116]

After the registration of the command response data is completed, the 'status' of the server command sheet is changed to 'response received' and indicated as thus (S126). Then, the fact that a response has been received is notified to a destination registered in the 'server command execution result notifying destination' (S127). With this notification, an application program such as that for generating the server command may be informed that a response to the generated command has been received, and may perform suitable processes according to the response.

[0117]

For example, when an application program for dealing with trouble arising in the HTTP client 11 generates a server command to acquire a sensor value from a temperature sensor of the HTTP client 11, this command is sent to the HTTP client, and the HTTP client 11 may send

back a command response including the requested sensor value. Then, the HTTP server 12 receiving this command response searches for the server command corresponding to this command response based on the command ID included in this command response, and registers this command response in association with its corresponding server command. Then, the application for dealing with the trouble that is registered as the command execution result notifying destination of this command is notified of the fact that a response to this command has been received. The application program receiving this notification may refer to the server command sheet and acquire the execution result of the generated command from the item 'output parameter' of this server command sheet.

After the processes up to step S127 are completed for the subjected part, the same processes starting from step S123 are repeated for a next part if such part exists.

[0118]

On the other hand, when it is determined in the step S123 that the subjected part corresponds to a client command, the XML text describing the part is analyzed and converted into data suitable for registration in the client command sheet (S128). Then, a client command sheet for this client command is created and this server command sheet containing the server command and its corresponding command ID is registered in the client command pool (S129). The content of the client command is registered in the items 'method name' and 'input parameter' of the client command sheet, and the command ID described in the part is registered in the item 'command ID' of this client command sheet. In the item 'client command notifying destination', reference information to the application program with which the method registered in the 'method name' is to be executed, for example, is registered, this reference information being determined based on correspondence information between a method and an application program, for example, that is provided beforehand. It is noted that the initial value of the 'status' corresponds to 'not processed', and the initial

value of the 'output parameter' corresponds to NULL.

[0119]

After the processes up to step S129 are completed for the subjected part, the same processes starting from step S123 are repeated for a next part if such part exists.

Thus, the process flow illustrated in Fig. 26 ends after the process sequence from step S123 to step S129 is performed on each of all the parts.

With the above, the processes pertaining to receiving the HTTP request are completed, and the process moves to the processes corresponding to steps S116 and onward of Fig. 25.

[0120]

Fig. 27 is a flowchart illustrating a more detailed process flow of steps S116 and onward of Fig. 25. The process step following step S127 or S129 of Fig. 26 corresponds to step S131 of this drawing.

According to this process flow, first, the CPU 31 of the HTTP server 12 collects from the server command pool 141 contents of the 'method name' and 'input parameter' of the server command sheets indicating 'not sent' for the 'status' as server commands to be sent to the HTTP client 11, and also collects contents of the 'command ID' as IDs of the collected server commands (S131). When 'not sent' is indicated as the 'status', this means that a command generated by the server command generating means 143 has not yet been sent to the HTTP client 11. Thus, commands to be sent to the HTTP client 11 may be extracted based on the information.

[0121]

Then, a process sequence corresponding to steps S132 to S134 is successively performed on each of the server commands collected in step S131. In this process sequence, first, a server command and its corresponding command ID that are subjected to the process are converted into an XML text in which information on the client command and the command ID are included in a SOAP body and a SOAP header, respectively (S132). A SOAP envelope corresponding to a part

describing the subject command is generated (S133). Then, the 'status' of the server command sheet describing the subject server command is changed from 'not sent' to 'waiting for response' (S134). When the 'status' is indicated as 'waiting for response', this means that the described command has been sent to the HTTP client 11.

[0122]

After the process sequence is performed on each of the collected server commands, the CPU 31 collects the contents of the 'output parameter' of client command sheets of which the 'status' is indicated as 'process completed' as command responses to client commands that are to be transmitted, and also collects the contents of the 'command ID' as IDs of the client commands corresponding to the collected responses (S135). When the 'status' is indicated as 'process completed', this means that the response corresponding to the client command that is generated by the client command execution result generating means 144 has not yet been sent to the HTTP client 11. Thus, the command responses to be transmitted to the HTTP client 11 may be extracted based on this information.

[0123]

Then, a process sequence corresponding to steps S136 to S138 is successively performed on each of all the command responses collected in step S135. The process sequence includes converting a subjected command response and the corresponding command ID collected therewith into an XML text that describes information on this command response and the corresponding command ID in a SOAP body and a SOAP header, respectively (S136), generating a SOAP envelope corresponding to a part describing the subjected command response (S137). The process sequence is identical to that of steps S132 and S133 except for the difference in the data items being subjected to the processes. Next, the 'status' of the client command sheet describing the subjected command response is changed to 'response made' (S138). When 'response made' is indicated as the 'status', this means that the command response has been sent to the HTTP client 11.

[0124]

After the completion of the processes described in above, the CPU 31 merges the parts generated in the steps S133 and S137, generates a multipart HTTP response as shown in Fig. 12 , and sends this to the HTTP client 11 (S139).

The changing of the 'status' in the steps S134 and S138 may be performed after the transmission of data is actually completed. As arranged in above, even when communication error occurs, commands to be transmitted and the command responses may be subjected for transmission again, whereby system reliability is improved.

[0125]

Herein, it is described that the parts to be transmitted are generated and merged together before they are transmitted, and the parts are received as a whole after which they are divided into parts for further processing; however, the present invention is not limited to this arrangement. For example, the parts may be successively transmitted each time a part is generated, or the parts may be successively received and processed accordingly each time a part is received, as described in the case of the HTTP client 11.

[0126]

In the following, processes pertaining to the execution of a client command are described.

Fig. 28 is a flowchart illustrating an example of such processes.

The process pertaining to the execution of a server command is performed after step S129 of Fig. 27. Namely, these steps may be performed after the client command is registered in the client command pool 42. The CPU 31 of the HTTP server 12 functions as the client command execution result generating means 144.

[0127]

In the process flow, first, an application program or some other suitable means for executing the client command is called based on the information corresponding to the 'client command notifying destination' of the client command sheet describing the registered

client command, and data corresponding to the 'method name' and 'input parameter' are handed over to the application so that the process pertaining to the server command is performed (S141). Although it is not shown in this flowchart, the process pertaining to the server command is executed by the CPU 31 using a different thread.

[0128]

After the process step is completed, the execution result is registered in the item 'output parameter' of the client command sheet (S142). At the same time, the 'status' of the client command sheet is changed to 'process complete' which indicates that the process is completed (S143). Then, the operation goes back to the process flow of Fig. 27.

By means of the process steps of Fig. 28, the client command may be executed and appropriate measures may be taken so that the execution result of the client command may be sent to the HTTP client 11 as the command response.

[0129]

As processes related to the execution of the client command, a process flow illustrated in Fig. 29 may be considered independently from the process flow of Fig. 27. The CPU 31 of the HTTP server 12 also functions as the client command execution result generating means 144.

In performing the process steps of Fig. 29, the CPU 31 initiates the process illustrated in the flowchart of Fig. 29.

[0130]

In the process of Fig. 29, it is judged whether a client command sheet of which the 'status' is indicated as 'waiting for process' exists at the client command pool (SY1), and if there is none, the process is held until such a client command sheet is added. Then, when it is found that such client command sheets exist, one of the command sheets is subjected for the process and the 'status' of its command sheet is changed to 'processing' (SY2).

Then, the process steps S141 to S143 are performed as similar in

Fig. 28 to execute the client command described in the subjected client command sheet, and the process is repeated returning to the step SY1 after these steps are completed.

It is noted that the above processes may be performed simultaneously using a plurality of threads as has been described for the HTTP client 11.

[0131]

By performing the above-described processes, each client command may be executed at an arbitrary timing and, thereby, even when the execution of one command takes a long time, the subsequent processes may not necessarily be affected by this delay. The execution results of these client commands are successively adapted for transmission to the HTTP client 11 as command responses in the order in which the execution of each command is completed.

In the above, the processes pertaining to the transfer of commands and command responses performed in the HTTP server 12 has been described.

[0132]

By implementing the above-described functions in the HTTP client 11 and HTTP server 12 so that they are able to perform the above-described processes, an operation request to be sent from a sender to a communication counterpart, and an operation response to an operation request received from the communication counterpart may be collectively sent to the communication counterpart in one bundle. In this way, it becomes unnecessary to perform individual negotiations to establish connections for the transmission of the operation request and the transmission of the operation response, respectively, and thereby, the communication overhead may be reduced and communication efficiency may be improved.

[0133]

The collective transmission in a bundle of the operation request and the operation response is realized by converting each of the operation request and the operation response into serialized data,

and further converting each data into a transmission message described in a structured language format. In this way, the operation request and operation response having different formats may be easily merged and they may be transmitted as one logical transmission content.

With this arrangement, the communication apparatus at the communication counterpart is able to receive an operation response to the operation request sent to the communication counterpart and an operation request from the communication counterpart in a batch, and easily divide the received content into individual messages to perform the appropriate processes depending on whether each message corresponds to an operation request or an operation response.

[0134]

In an embodiment where one communication apparatus is arranged to send a communication request and the communication apparatus at the other end (communication counterpart) is arranged to send messages such as an operation request to the sender of the communication request as a communication response to this communication request, the transmission/reception of operation requests and operation responses may be performed smoothly even in a communication system in which the communication apparatus sending the communication request (communication client) is stationed within a firewall. Also, in this embodiment, since the communication request and communication response correspond to each other, timing management in the communication level may be easy.

[0135]

By arranging the above communication apparatus sending a communication request to periodically send the communication request to its communication counterpart in this embodiment, a delay in the transmission of information from outside to be transmitted inside the firewall for a long period of time can be prevented.

By implementing a client command pool and a server command pool, the operation requests and operation, responses generated by various means such as application programs may be accumulated in theses pools

and, thus, the operation requests and operation responses may be generated without due consideration to their transmission timing to the communication counterpart. Thereby, the processes performed by the various application programs and/or other suitable means may be simplified and, in turn, the design and development of these application programs may also be simplified.

[0136]

By implementing gathering means for reading from the pools the operation requests and operation responses that are to be transmitted to the communication counterpart, information to be transmitted may be prevented from being left out in the transmission.

Distributing means may be implemented for dividing the received operation requests and operation responses and storing them in the respective pools. In this way, received information may be accumulated in the pools and, thereby, the execution of an operation pertaining to a received operation request and/or the execution of processes after receiving an operation response may be performed without due consideration to a reception timing from the communication counterpart. Therefore, the processes performed by the various application programs and/or other suitable means may be simplified and, in turn, the design and development of these application programs may also be simplified.

[0137]

By assigning identification information such as an ID to a generated operation request, and storing or transmitting the operation request in association with this identification information, and also storing or transmitting an operation response in association with the identification information of an operation request for this operation response, correspondence information between an operation request and an operation response may be easily recognized based on the identification information even in a case where a plurality of operation requests and operation responses are included in one message.

By assigning priority information to the operation requests and executing and sending responses for these requests in due order according to the priority information, an operation requiring urgency may be executed with priority and also, a response to this operation request may be sent with priority.

[0138]

[Reference example applying SMTP: Figs. 30 to 35]

In the following, a reference example in which SMTP is used as the communication protocol is described. It is noted that many common points exist between this reference example and the previously-described embodiment using HTTP as the communication protocol. Thus, parts of the description of this reference example that are identical to those of the embodiment using HTTP are omitted or simplified, and an emphasis is put on features of this reference example that are different from those of the embodiment using HTTP.

Fig. 30 is a block diagram showing an exemplary configuration of a communication system implementing the above reference example using SMTP.

[0139]

As shown in Fig. 30, the communication system includes a LAN_A connecting a communication apparatus A and a mail server A', a LAN_B connecting a communication apparatus B and a mail server B', and the Internet 13 connecting the LAN_A and LAN_B via a firewall A of the LAN_A and a firewall B of the LAN_B. A mail server A and a mail server B are provided in the LAN_A and LAN_B, respectively, at positions enabling access from outside via the respective firewalls. It is noted that the communication apparatus A corresponds to a first communication apparatus, and the communication apparatus B corresponds to a second communication apparatus.

[0140]

In a communication using SMTP, the transfer of information between the communication apparatus A and the communication apparatus B is realized by electronic mail. Specifically, for example, when

information is to be sent from the communication apparatus B to the communication apparatus A, the communication apparatus B outputs an electronic mail addressed to the communication apparatus A, which is first sent to the mail server B', as is indicated by the dashed line arrows in Fig. 30. Then, the electronic mail is transferred via the mail servers B', B, and A, respectively, before being transferred to the mail server A' to which the communication apparatus A has direct access. Although illustration in a drawing is omitted, transfer is performed via another mail server between the mail server B and the mail server A.

[0141]

Then, as indicated by the dash-dotted line arrow, the communication apparatus A may periodically access the mail server A' to receive electronic mail addressed to thereto. In this way, information transmission from the communication apparatus B to the communication apparatus A is completed. On the other hand, in transmitting information from the communication apparatus A to the communication apparatus B, a reverse procedure of the above-described procedure may be performed. In other words, the communication apparatus A and the communication apparatus B operate in a similar manner with respect to information transmission. It is noted that the mail server A' and mail server B' do not necessarily have to be arranged, and the communication apparatus A may communicate directly with the mail server A and the communication apparatus B may communicate directly with the mail server B. Also, a mail server for receiving mails may be different from a mail server for mail transmission.

[0142]

In such a system, each LAN implements a mail server at a position that allows access from outside so that electronic mail may be sent thereto via the firewall.

In the reference example, the communication apparatus A and the communication apparatus B are able to transmit information to/from each other even though they do not communicate through direct

negotiation.

[0143]

As with the first and second communication apparatuses of Fig. 1, the communication apparatus A and communication apparatus B implement application programs for controlling and managing each other. Using RPC, each communication apparatus is arranged to send an 'operation request' calling for the execution of the process for a method of an application program implemented at the other side, and acquire an 'operation response' corresponding to the execution result of the requested process.

[0144]

Fig. 31 shows relations between the operation request and the operation response. As is shown, in the operation request level, the communication apparatus A and the communication apparatus B operate in a similar manner as with the HTTP client 11 and the HTTP server 12. However, in the case of using SMTP, the communication apparatus A and the communication apparatus B operate in a similar manner even in the communication level, this being different from the relation of the HTTPs.

[0145]

Fig. 32 shows an exemplary communication sequence in this communication system.

As described earlier, in this communication system, communication between the communication apparatus A and the communication apparatus B is performed using electronic mail. This electronic mail includes information on a sender and a destination, and a reply may be sent back to the sender from the destination. However, in this case, the first mail and the reply mail are independent from each other; that is, in this example there is no relation similar to the communication request-communication response relation found in the embodiment using HTTP.

Thus, either of the two communication apparatuses is able to initiate the communication, and the electronic mail does not

necessarily have to be sent back and forth. In the following example, a case will be described in which a total of three electronic mails are transmitted/received back and forth starting with the communication apparatus A sending an electronic mail to the communication apparatus B.

[0146]

In these electronic mails, an operation request (command) to a destination (address) and an operation response (command response or simply 'response') to the command received from the destination are described. This case applies whether the sender corresponds to the communication apparatus A or the communication apparatus B.

Thus, for example, communication apparatus A command a may be described in electronic mail x to be transmitted to the communication apparatus B, and subsequently, the corresponding command response from the communication apparatus B may be described in electronic mail y to be sent to the communication apparatus A. Also, communication apparatus B command c may be described in the electronic mail y and, subsequently, the corresponding command response from the communication apparatus A may be described in electronic mail z to be sent to the communication apparatus B.

[0147]

It is noted that any number (may be 0) of operation requests and operation responses to the operation request may be described in one electronic mail. Further, the content described in one electronic mail corresponds to one message and is logically sent as one batch. In this way, the number of electronic mails for transmitting information can be reduced so that the communication overhead is reduced and communication efficiency is improved.

It is also noted that a command response may be described in a first electronic mail sent after the reception of the corresponding command or in an electronic mail sent thereafter as in the example of Fig. 6 using HTTP.

[0148]

Next, in the following, functional configurations and process steps for realizing the processes of merging the commands and command responses and separating these parts in the communication apparatus A and communication apparatus B will be described. It is noted that the hardware configurations of these apparatuses may be identical to the configuration illustrated in Fig. 7 for the HTTP client 11 and HTTP server 12.

[0149]

Fig. 33 is a functional block diagram illustrating a functional configuration of the communication apparatus A for performing the processes pertaining to the commands and command responses corresponding to Fig. 8.

As is shown, the communication apparatus A includes a communication apparatus A command pool 51, a communication apparatus B command pool 52, communication apparatus A command generating means 53, and communication apparatus B command execution result generating means 54, which correspond to the client command pool 41, the server command pool 42, the client command generating means 43, and the server command execution result generating means 44, respectively, as shown in Fig. 8. The differences in the names of these corresponding functions are due to the differences in the names of the apparatuses.

[0150]

The transmission message collection means 45 and the receiving message distribution means 48 correspond to the means with the same names as shown in Fig. 8. It is noted that in this example, a data format corresponding to SMTP is different from the case shown in Fig. 8. However, the individual SOAP messages corresponding to the commands and command responses described are identical to those used in the embodiment of Fig. 8.

The mail transmitting means 56 and the mail receiving means 57 correspond to transmitting means and receiving means, but are different from the HTTP request transmitting means 46 and the HTTP response receiving means 47 since the protocol being used for

transmission and reception of data in the two examples is different.

[0151]

The mail transmitting means 56 has the function of generating an electronic mail addressed to the communication apparatus B that includes the outgoing message generated by the outgoing message gathering means 45, and sending this to the mail server A'. After this, the communication apparatus A does not interfere with the transmission of this electronic mail. As in the embodiment of Fig. 8, any number of outgoing messages may be included in one electronic mail, and the outgoing message may correspond to a command response or a communication apparatus A command. The outgoing messages corresponding to a command response and outgoing messages corresponding to a communication apparatus A command may be intermingled on an arbitrary basis.

[0152]

The mail receiving means 57 has the function of checking to see whether there are any newly arrived mails, and receiving the newly arrived mails if they exist. The electronic mail to be received may include any number of receiving messages describing a communication apparatus B command and its associated command ID, and receiving messages describing a response to a communication apparatus A command and its associated command ID, that are intermingled with each other on an arbitrary basis.

[0153]

Fig. 34 shows an exemplary electronic mail to be sent to the communication apparatus B from the communication apparatus A having the above-described functions.

This electronic mail, like the HTTP request of Fig. 11, has a body portion that describes a multipart message using MIME (Multipurpose Internet Mail Extension). In each part making up the multipart message a SOAP envelope is embedded. Thus, the body portion of this electronic mail is identical to the HTTP body of the HTTP request.

[0154]

However, the header portion of this electronic mail is different from the HTTP header in that information corresponding to items 'From' for indicating the sender address of the electronic mail, 'To' for indicating the destination address, and 'Subject' for indicating the title, for example, are described therein.

As for the electronic mail to be sent to the communication apparatus A from the communication apparatus B, the information content described in the items 'From' and 'To' are exchanged.

Also, the contents of the SOAP envelopes described in the electronic mails are identical to those described in the HTTP request or HTTP response. However, in the entity header portion of each part, different information may be described since the data encoding method used in this example is different from that used in the embodiment using HTTP.

[0155]

In the following, processes executed in the communication apparatus A having the above-described configurations and functions will be described with reference to flowcharts shown in Figs. 35 and 36. It is noted that the processes shown in these flowcharts are executed by a CPU that is implemented in the communication apparatus A and executes suitable control programs.

[0156]

Fig. 35 is a flowchart illustrating a basic operation flow of processes executed upon message transmission.

First, the CPU of the communication apparatus A starts the process flow of Fig. 35 at an appropriate timing for the outgoing message gathering means 45 to undertake a readout of communication apparatus A commands and/or command responses.

Then, a communication apparatus A command gathering process is performed (S51). This corresponds to a process of gathering from the communication apparatus A command pool 51 communication apparatus A commands to be sent to the communication apparatus B, and also includes a process of generating SOAP envelope parts based on the gathered data.

[0157]

Then, a communication apparatus B command execution result gathering process is performed (S52). This corresponds to a process of gathering from the communication apparatus B command pool 52 command responses to be sent to the communication apparatus B, and also included a process of generating SOAP envelope parts based on the gathered data.

Then, the parts generated in the processes of steps S51 and S52 are merged so that an electronic mail containing these parts is generated (S53). This electronic mail is then addressed to the communication apparatus B and sent thereto (S54) and, hereby, the message transmission process is completed.

[0158]

In executing the above-described processes, the CPU31 functions as the transmission message collection means 45 in steps S51 and S52, and as the mail transmitting means 56 in steps S53 and S54.

It is noted that these processes correspond to steps S11 to S14 shown in Fig. 17. However, in the case of using SMTP, the transmission of electronic mail and the reception of electronic mail each correspond to individual processes and a relation such as that between the communication request and the communication response in the data transmission/reception between the HTTP client 11 and HTTP server 11 does not exist in this example. Thus, the transmission process ends here, and another separate process shown in Fig. 36 is performed in receiving the electronic mail.

[0159]

Fig. 36 is a flowchart illustrating a basic operation flow of a message receiving process.

The CPU of the communication apparatus A periodically accesses the mail server A' and starts the process illustrated in the flowchart of Fig. 36 when it detects a newly arrived electronic mail addressed to the communication apparatus A.

In this process, first, the CPU receives the newly arrived

electronic mail (S61). Then, the body (text) of the received electronic mail is divided into parts (S62). This corresponds to a process of dividing the body into components that are separated from each other by a 'MIME_boundary'.

[0160]

Then, a sequence of processes corresponding to steps S63 to S65 is successively performed for each of the divided parts. In this process sequence, first, it is determined whether a part subjected to the process describes a communication apparatus B command (S63). Then, if it is determined that the part describes a communication apparatus B command, a communication apparatus B command registration process is performed (S64). On the other hand, if it is determined that the part does not describe a communication apparatus B command, this means that the part describes a response to a communication apparatus A command and, thereby, a response notification process is performed (S65).

[0161]

After performing either step S63 or S64, the process goes back to step S63, and the process sequence is repeated for a next part that is subjected to the process. Thus, the process sequence of steps S63 to S65 is performed for each of the divided parts, after which the message receiving process of Fig. 36 ends.

In executing the above-described processes, the CPU31 functions as the mail receiving means 57 in step S61, and as the received message distribution means 48 in steps S62 to S65.

These processes correspond to steps S15 to S19 of Fig. 17.

[0162]

It is also noted that the processes relating to the execution of a communication apparatus B command in the communication apparatus A are identical to the processes relating to the execution of the server command described with reference to Figs. 20 and 21.

In the above, the processes relating to the transmission of a command and a command response performed in the communication

apparatus A have been described.

As for the functions and processes of the communication apparatus B, the concepts of the communication apparatus A command and the communication apparatus B command may be switched in the above descriptions of the communication apparatus A.

[0163]

As described above, the present invention may be applied to communications performed by using a protocol in which transmission of information does not necessarily correspond to receiving information as a response to this transmission. Even in such case, an operation request to be sent from a sender to a communication counterpart, and an operation response to an operation request that has been received from the communication counterpart may be collectively sent to the communication counterpart in one batch so that the operation request and the operation response do not have to be transmitted in separate units. In this way, the communication overhead may be reduced and communication efficiency may be improved.

Also, by using electronic mail for data transmission and receiving, information may be easily transferred within the firewalls.

[0164]

[Modification Example of Embodiment and Reference Examples: Fig. 37]

In the following, an exemplary modification of the above-described embodiment and reference example is described.

In the above described embodiment and reference, the present invention is described by illustrating an exemplary communication system that is made up of two communication apparatuses; however, this has been done to simplify the description and the present invention is not limited to these examples. For example, the present invention may be applicable to a communication system made up of a greater number of communication apparatuses or a communication apparatus included in such communication system.

For example, another exemplary communication system that uses HTTP may be applied to a communication system shown in Fig. 37.

[0165]

The communication system configures a remote management system for managing a target managing apparatus 100 by means of a managing apparatus 102 by establishing a connection between the managing apparatus 102 having communication functions and the target managing apparatus 100 having also communication functions via a network including the Internet 13.

The target managing apparatus 100 may correspond to various types of electronic apparatuses having communication functions including, for example, a printer, a scanner, a copier, a digital multi functional imaging apparatus implementing more than one of the above functions, a network appliance, a vending machine, medical equipment, a power source apparatus, an air conditioning system, measuring systems for gas, water, or electricity, a computer that can be connected to a network.

[0166]

The remote management system includes an intermediate device 101 which corresponds to a communication apparatus for mediating the communication between the managing apparatus 102 and the target managing apparatus 100. Thus, the managing apparatus 102 and the target managing apparatus 100 communicate with each other via the intermediate device 101.

The intermediate device 101 and the target managing apparatus 100 may be arranged in various structures in configurations thereof according to the environment in which those apparatuses are used. For example, in environment A shown in Fig. 37, the target managing apparatuses 100a and 100b are arranged under the control of the intermediate device 101a which is capable of establishing a connection with the managing apparatus 102 using HTTP. On the other hand, in environment B, four units of the target apparatus 100 are provided and, thereby, the load may be too large for one intermediate device 101 to subordinate the managed apparatuses.

[0167]

For this reason, the intermediate device 101b which may establish a connection with the managing apparatus 102 1 using HTTP subordinates not only the target managing apparatuses 100c and 100d, but also the other intermediate device 101c. Further, the intermediate device 101c subordinates the target apparatuses 100e and 100f. Information generated from the managing apparatus 102 to conduct remote management on the managed apparatuses 100e and 100f passes through the intermediate device 101b, and then the intermediate device 101c, which corresponds to a subordinate node of the intermediate device 101b, to reach the target apparatus 100e or 100f. Also, for security reasons, firewall 14 is implemented in each environment.

[0168]

By handling the intermediate device 101 as the HTTP client and the managing apparatus 102 as the HTTP server, the present invention may be applied to the remote management system.

In the case, when the managing apparatus 102 is to send a command to the target apparatus 100, the managing apparatus 102 may first send a command 'send command to the target managing apparatus 100', and the intermediate device 101 may be arranged to send the command to the target managing apparatus 100 as an operation corresponding to the command received from the managing apparatus 102. Alternatively, the managing apparatus 102 may designate any one of the target managing apparatuses 100 as a destination of a command and then send this command to the intermediate device 101, and the intermediate device 101 may be arranged to send a command received from the managing apparatus 102 that is addressed to a destination other than itself to the designated destination.

[0169]

When transmission/reception of commands and command responses is performed among three or more nodes, information on the sender and destination of the command may be included in a message corresponding to the command or command response so that the sender and destination of the command may be determined, and preferably, this information

is described and managed in a command sheet as well.

[0170]

By implementing a mail server in this system, the reference example using SMTP may also be applied to this system.

The present invention is not limited to these embodiments, and further variations and modifications may be made without departing from the scope of the present invention. For example, the client command sheet and server command sheet respectively registered in the client command pool 41 and the server command pool 42 may be described as XML formatted documents.

A limit may be imposed on the amount of information contained in the command response to be transmitted/received. Especially, by limiting the amount of receiving information of the command, usage amount of memory may be contained when the receiving side is an apparatus having limited memory capacity.

[0171]

In the above described embodiment and reference example, SOAP is used as the upper protocol for realizing RPC and RPC is realized with an application directly operating the pool. However, other protocols such as CORBA (Common Object Request Broker Architecture) or JAVA (Registered Trademark) RMI (Remote Method Invocation) may be used as well between the application and the pool.

In other words, according to the above embodiment and reference example, the exchange of a command and a response to the command between the HTTP client 11 and the HTTP server 12 is realized by a SOAP message described in XML; however, the present invention is not limited to this arrangement and data may be described in other formats as well.

[0172]

In the above-described embodiment and reference examples, unique protocols are used in addition to protocols according to the SOAP standard so that the SOAP envelopes included in the HTTP request or HTTP response may be handled as independent parts. Using the SOAP attachment covered by the SOAP standard protocol, the SOAP envelope

corresponding to the first part of the HTTP response may be arranged to contain links to subsequent SOAP envelopes corresponding to the second part and onward so that they are associated when handed down. This arrangement may similarly be applied to the reference example using electronic mail and SMTP.

[0173]

Further, a case has been described, wherein HTTP is used for data transmission as the subordinate protocol with respect to the upper protocol such as SOAP. However, other protocols such as FTP (File Transfer Protocol) may also be used as the subordinate protocol.

The configuration of the communication system according to the present invention is not limited to the examples described in above.

[0174]

The programs according to the present invention correspond to application programs for causing a computer to function as a communication apparatus such as the HTTP client 11, the HTTP server 12, the communication apparatus A or B that is capable of communicating with another communication apparatus as a communication counterpart, and by executing such programs in a computer, the various effects described above may be obtained.

[0175]

Such programs may be stored in advance in storage means provided in the computer such as a ROM or an HDD. Alternatively, the programs may be recorded on a recording medium such as a CD-ROM, or a nonvolatile recording medium (memory) such as a flexible disk, a SRAM, an EEPROM, or a memory card. In this case, the programs recorded on the memory may be installed in the computer so that the CPU may execute the programs, or the CPU may be arranged to read the programs as well as executing them. In this way, the various processes described above may be realized.

As another alternative, the programs may be downloaded from an external apparatus that is connected to a network and has a recording medium on which the program is recorded, or stores the programs in

its storage means, so that the programs are executed.

[Utility in the Industry]

[0176]

As described in above, by using the communication server, the control method of the communication server, the program, and the recording medium of the present invention, communication efficiency may be improved in a communication system in which a plurality of communication apparatuses for transmitting/receiving a communication request and a communication response to the communication request transmit/receive operation requests and operation responses to the received operation requests for each other.

Therefore, a communication system bearing a small communication load may be configured by applying the invention to such communication system.

[Brief Description of the Drawings]

[0177]

[Fig. 1]

Fig. 1 is a diagram illustrating an exemplary configuration of a communication system of the invention using the communication apparatus of the present invention.

[Fig. 2]

Fig. 2 is a diagram illustrating a relationship of an operation request and an operation response according to the communication system shown in Fig. 1.

[Fig. 3]

Fig. 3 is a diagram illustrating an exemplary configuration of a communication system in a case when HTTP is used as a communication protocol.

[Fig. 4]

Fig. 4 is a diagram illustrating a relationship of an operation request and an operation response in the communication system shown in Fig. 3.

[Fig. 5]

Fig. 5 is a diagram illustrating an example of a communication sequence in the communication system shown in Fig. 3.

[Fig. 6]

Fig. 6 is a diagram illustrating another example of the above communication sequence.

[Fig. 7]

Fig. 7 is a diagram illustrating an exemplary configuration of hardware of the HTTP client and HTTP server shown in Fig. 3.

[Fig. 8]

Fig. 8 is a functional block diagram illustrating functional configuration for performing a process related to a command and a command response in functions of the HTTP client shown in Fig. 3.

[Fig. 9]

Fig. 9 is a diagram illustrating an example of data configuration in a client command sheet to be stored in the client command pool shown in Fig. 8.

[Fig. 10]

Fig. 10 is a diagram illustrating an example of data configuration in a server command sheet to be stored in the server command pool shown in Fig. 8.

[0178]

[Fig. 11]

Fig. 11 is a diagram illustrating an example of a HTTP request which the HTTP client shown in Fig. 3 transmits to the HTTP server.

[Fig. 12]

Fig. 12 is a diagram illustrating an example of a HTTP response which the HTTP client shown in Fig. 3 receives from the HTTP server.

[Fig. 13]

Fig. 13 is a diagram illustrating an example of a part in which a client command is described.

[Fig. 14]

Fig. 14 is a diagram illustrating an example of a part in which a response to a client command is described.

[Fig. 15]

Fig. 15 is a diagram illustrating an example of a part in which a server command is described.

[Fig. 16]

Fig. 16 is a diagram illustrating an example of a part in which a response to a server command is described.

[Fig. 17]

Fig. 17 is a flowchart illustrating a functional operation of a message collection and distribution process in the HTTP client shown in Fig. 3.

[Fig. 18]

Fig. 18 is a flowchart illustrating in more detail a process of parts in the steps S11 to S14 of Fig. 17.

[Fig. 19]

Fig. 19 is a flowchart illustrating in more detail a process of parts in the steps on and after S15 of Fig. 17.

[Fig. 20]

Fig. 20 is a flowchart illustrating an example of a process related to execution of a server command in the HTTP client shown in Fig. 3.

[0179]

[Fig. 21]

Fig. 21 is a flowchart illustrating another example of the above process.

[Fig. 22]

Fig. 22 is a functional block diagram illustrating a configuration of a function for performing a process related to a command and a command response in the functions of the HTTP server shown in Fig. 3.

[Fig. 23]

Fig. 23 is a diagram illustrating an example of a data structure in a server command sheet to be stored in the server command pool shown in Fig. 22.

[Fig. 24]

Fig. 24 is a diagram illustrating an example of a data structure in a client command sheet to be stored in the client command pool shown in Fig. 22.

[Fig. 25]

Fig. 25 is a flowchart illustrating a fundamental operation of a message collection and distribution process in the HTTP server shown in Fig. 3.

[Fig. 26]

Fig. 26 is a flowchart illustrating in more detail a process of the parts in the steps S111 to S115 of Fig. 25.

[Fig. 27]

Fig. 27 is a flowchart illustrating in more detail a process of the parts in the steps on and after S116 of Fig. 25.

[Fig. 28]

Fig. 28 is a flowchart illustrating an example of a process related to execution of a client command in the HTTP server shown in Fig. 3.

[Fig. 29]

Fig. 29 is a flowchart illustrating another example of the above process.

[Fig. 30]

Fig. 30 is a diagram illustrating a configuration example of a communication system of reference example in a case when SMTP is used as a communication protocol.

[0180]

[Fig. 31]

Fig. 31 is a diagram illustrating a relationship of an operation request and an operation response in the communication system shown in Fig. 30.

[Fig. 32]

Fig. 32 is a diagram illustrating an example of a communication sequence in the communication system shown in Fig. 30.

[Fig. 33]

Fig. 33 is a functional block diagram corresponding to Fig. 8,

illustrating a configuration of a function for performing a process related to a command and a command response in the functions of the communication apparatus A shown in Fig. 30.

[Fig. 34]

Fig. 34 is a diagram illustrating an example of an electronic mail which the communication apparatuse A sends to the communication apparatus B shown in Fig. 30.

[Fig. 35]

Fig.35 is a flowchart illustrating a fundamental operation of a process upon transmitting a message in the communication apparatus A shown in Fig. 30.

[Fig. 36]

Fig. 36 is a flowchart illustrating a flowchart of a fundamental operation of a process upon transmitting a message as well.

[Fig. 37]

Fig. 37 is a diagram illustrating an example of another configuration of the communication system of the present invention.

[Description of Numerals]

[0181]

1 :First communication apparatus 2 :Second communication apparatus
10 :Network 11 :HTTP client
12 :HTTP server 13 :Internet
14 :Firewall 31 :CPU
32 :ROM 33 :RAM
34 :SD card 35 :NIC
41, 142 :Client command pool
42, 141 :Server command pool
43 :Client command generation means
44 :Server command execution result generation means
45, 145 :Transmission message collection means
46 :HTTP request transmission means
47 :HTTP response receiving means
48, 148 :Receiving message distribution means

51 :Communication apparatus A command pool

52 :Communication apparatus B command pool

53 :Communication apparatus A command generation means

54 :Communication apparatus B command execution result generation means

56 :Mail transmission means 57 :Mail receiving means

100 :Target managing apparatus 101 :Intermediate device

102 :Managing apparatus 143 :Server command generation means

144 :Client command execution result generation means

146 :HTTP response transmission means

147 :HTTP request receiving means

[Name of Document] Drawings

[Name of Document] Abstract of the Disclosure

[Abstract]

[Objectives of the Invention] To improve communication efficiency when a plurality of communication apparatuses transmitting/receiving communication requests and communication responses to the communication requests transmit/receive operation requests and operation responses to the received operation requests for each other.

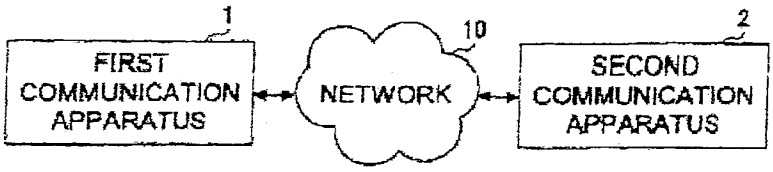
[Means for Achieving the Objectives]

In an HTTP server 12, an operation response to an operation request transmitted to an HTTP client and an operation request from the HTTP client are received in one batch by an HTTP request receiving means 147, and are divided and stored by an receiving message distribution means 148. Also, an operation request to be transmitted to the HTTP client and an operation response to an operation request from the HTTP client are collected by a transmission message collection means 145, and are transmitted by an HTTP response transmission means 146 in one batch. Then, when an operation request from the HTTP client is issued, a client command execution result generation means 144 executes an

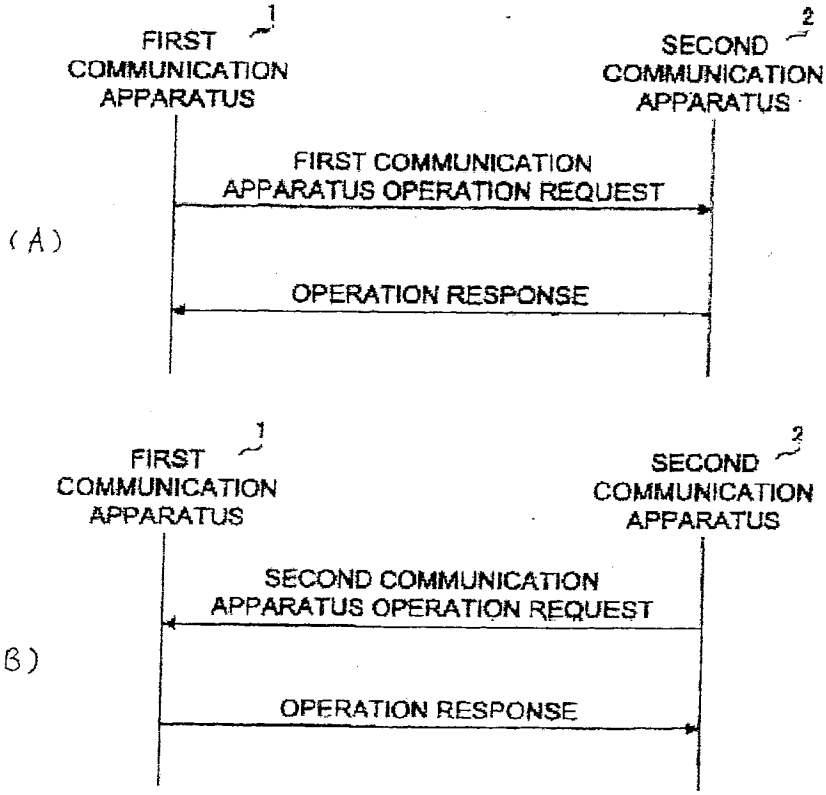
operation related to the operation request and generates an operation response.

[Selected Drawing] Fig. 22

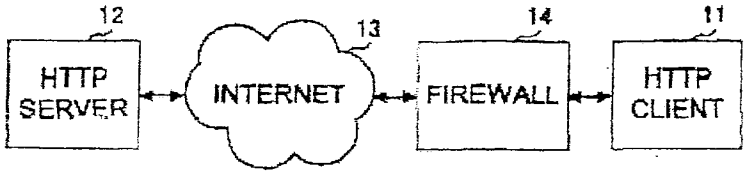
[Fig. 1]



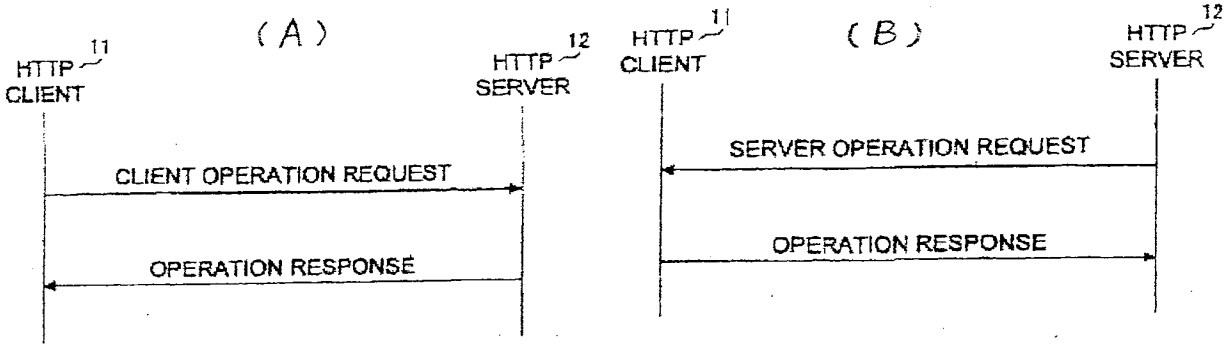
[Fig. 2]



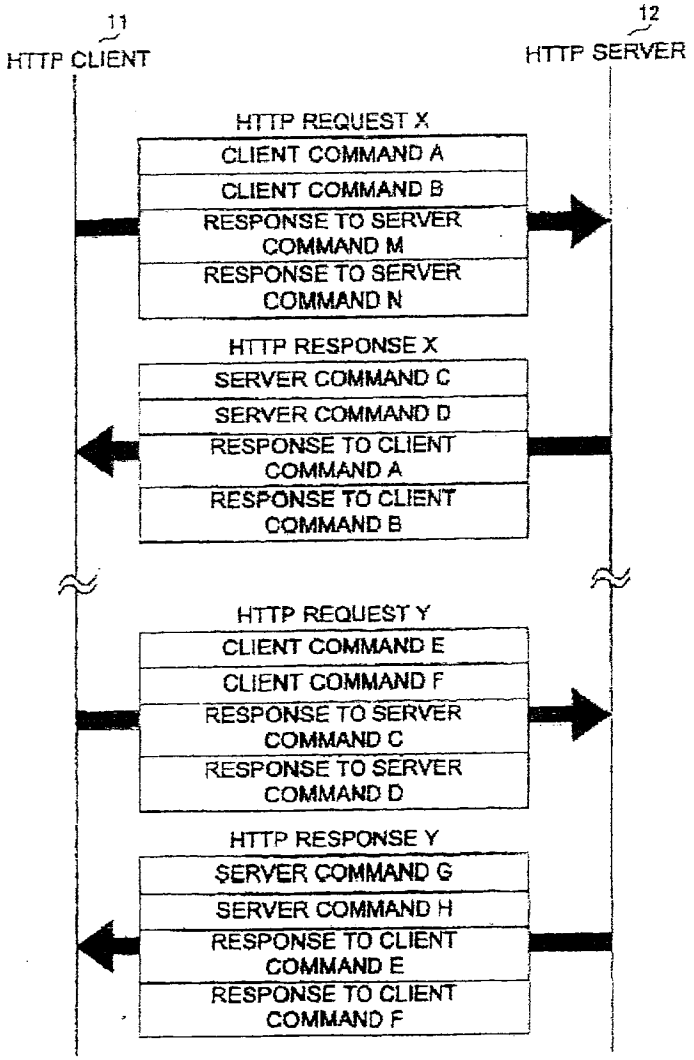
[Fig. 3]



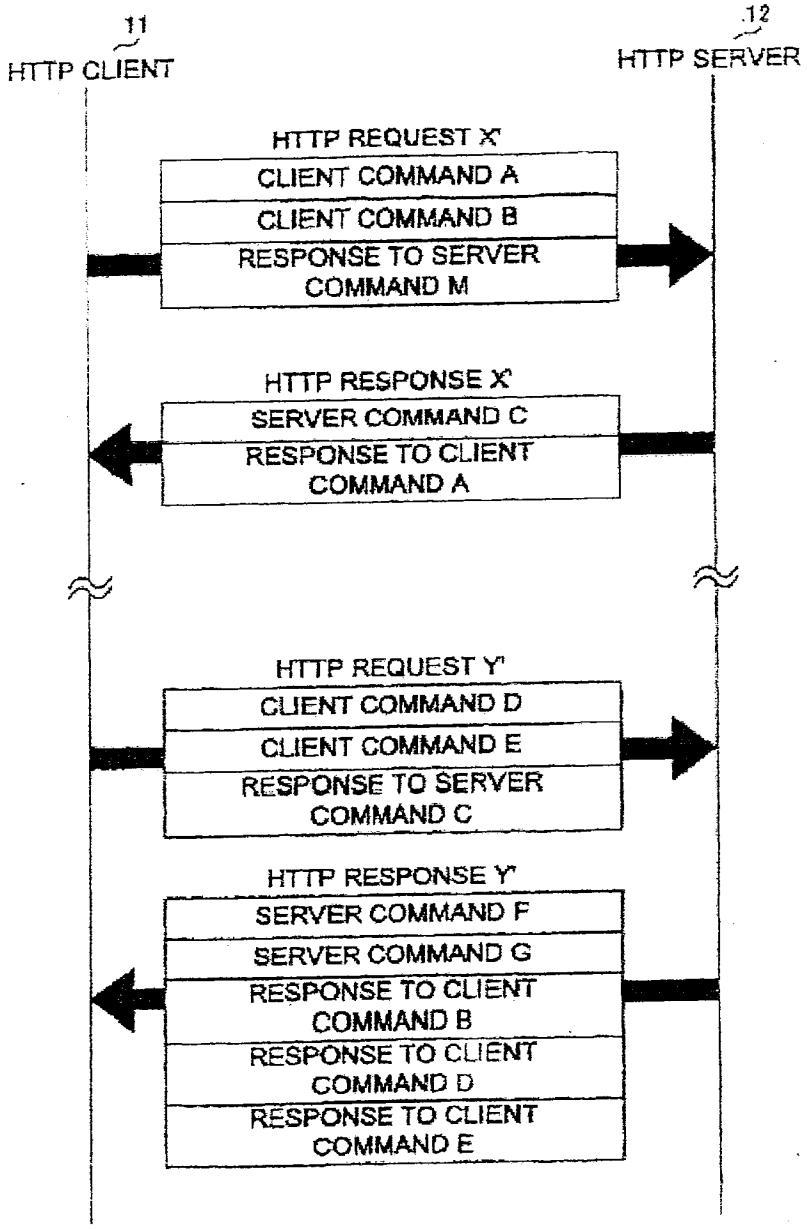
[Fig. 4]


(A)
HTTP CLIENT
11
HTTP SERVER
12
CLIENT OPERATION REQUEST
OPERATION RESPONSE
(B)
HTTP CLIENT
11
HTTP SERVER
12
SERVER OPERATION REQUEST
OPERATION RESPONSE

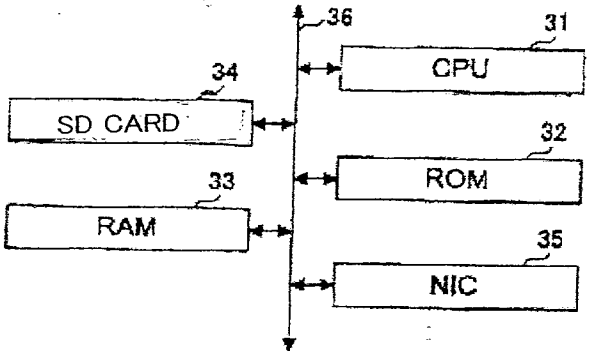
[Fig. 5]


HTTP CLIENT
11
HTTP SERVER
12
HTTP REQUEST X
CLIENT COMMAND A
CLIENT COMMAND B
RESPONSE TO SERVER COMMAND M
RESPONSE TO SERVER COMMAND N
HTTP RESPONSE X
SERVER COMMAND C
SERVER COMMAND D
RESPONSE TO CLIENT COMMAND A
RESPONSE TO CLIENT COMMAND B
HTTP REQUEST Y
CLIENT COMMAND E
CLIENT COMMAND F
RESPONSE TO SERVER COMMAND C
RESPONSE TO SERVER COMMAND D
HTTP RESPONSE Y
SERVER COMMAND G
SERVER COMMAND H
RESPONSE TO CLIENT COMMAND E
RESPONSE TO CLIENT COMMAND F

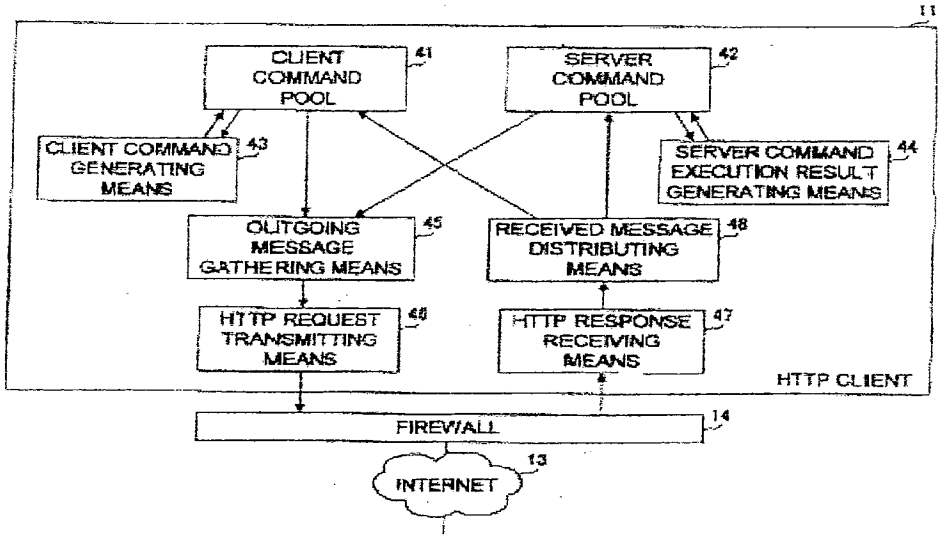
[Fig. 6]


11
HTTP CLIENT
12
HTTP SERVER
HTTP REQUEST X'
CLIENT COMMAND A
CLIENT COMMAND B
RESPONSE TO SERVER COMMAND M
HTTP RESPONSE X'
SERVER COMMAND C
RESPONSE TO CLIENT COMMAND A
HTTP REQUEST Y'
CLIENT COMMAND D
CLIENT COMMAND E
RESPONSE TO SERVER COMMAND C
HTTP RESPONSE Y'
SERVER COMMAND F
SERVER COMMAND G
RESPONSE TO CLIENT COMMAND B
RESPONSE TO CLIENT COMMAND D
RESPONSE TO CLIENT COMMAND E

[Fig. 7]



[Fig. 8]



[Fig. 9]

CLIENT COMMAND SHEET *IN HTTP CLIENT*

COMMAND ID
METHOD NAME (e.g., TROUBLE NOTIFICATION)
INPUT PARAMETER (e.g., TROUBLE CONTENT)
STATUS (INITIAL VALUE: NOT SENT)
CLIENT COMMAND EXECUTION RESULT NOTIFYING DESTINATION
OUTPUT PARAMETER (BLANK UNTIL RESPONSE IS ACQUIRED)

[Fig. 10]

SERVER COMMAND SHEET IN HTTP CLIENT

COMMAND ID

METHOD NAME (e.g., COUNTER ACQUISITION)

INPUT PARAMETER

STATUS (INITIAL VALUE: NOT SENT)

OUTPUT PARAMETER
(BLANK UNTIL PROCESS IS COMPLETED)

SERVER COMMAND NOTIFYING DESTINATION

[Fig. 11]

HTTP REQUEST

```
POST /aea HTTP/1.1
Content-Type:multipart/mixed;boundary=MIME_boundary
Content-Length:nnnn

--MIME_boundary                                          ┐
Content-Type:text/xml;charset=UTF-8                      │
Content-Transfer-Encoding:8bit                           │
SOAPAction:"SOAP Action URI"                             │
X-SOAP-Type:Request                                      │ PART 1
                                                         │
<s:Envelope>                                             │
   <--SOAP Request-->                                    │
</s:Envelope>                                            ┘

--MIME_boundary                                          ┐
Content-Type:text/xml;charset=UTF-8                      │
Content-Transfer-Encoding:8bit                           │
X-SOAP-Type:Response                                     │ PART 2
                                                         │
<s:Envelope>                                             │
   <--SOAP Response-->                                   │
</s:Envelope>                                            ┘

--MIME_boundary                                          ┐
Content-Type:text/xml;charset=UTF-8                      │
Content-Transfer-Encoding:8bit                           │
X-SOAP-Type:Response                                     │ PART 3
                                                         │
<s:Envelope>                                             │
   <--SOAP Response-->                                   │
</s:Envelope>                                            ┘

--MIME_boundary                                          ┐
Content-Type:text/xml;charset=UTF-8                      │
Content-Transfer-Encoding:8bit                           │
X-SOAP-Type:Response                                     │ PART 4
                                                         │
<s:Envelope>                                             │
   <--SOAP Response-->                                   │
</s:Envelope>                                            ┘

--MINE_boundary--
```

[Fig. 12]

HTTP RESPONSE

```
HTTP/1.1  200  OK
Content-Type:multipart/mixed;boundary=MIME_boundary
Content-Length:nnnn

--MIME_boundary
Content-Type:text/xml;charset=UTF-8
Content-Transfer-Encoding:8bit
SOAPAction:"SOAP Action URI"
X-SOAP-Type:Requset

<s:Envelope>
   <--SOAP Request-->
</s:Envelope>

--MIME_boundary
Content-Type:text/xml;charset=UTF-8
Content-Transfer-Encoding:8bit
X-SOAP-Type:Response

<s:Envelope>
   <--SOAP Response-->
</s:Envelope>

--MIME_boundary
Content-Type:text/xml;charset=UTF-8
Content-Transfer-Encoding:8bit
X-SOAP-Type:Response

<s:Envelope>
   <--SOAP Response-->
</s:Envelope>

--MIME_boundary
Content-Type:text/xml;charset=UTF-8
Content-Transfer-Encoding:8bit
X-SOAP-Type:Response

<s:Envelope>
   <--SOAP Response-->
</s:Envelope>

--MINE_boundary--
```

PART 1

PART 2

PART 3

PART 4

[Fig. 13]

EXEMPLARY PART
DESCRIBING CLIENT COMMAND

```
Content-Type: text/xml; charset=UTF-8
Content-Transfor Encoding: 8bit
X-SOAP-Type : Request
SOAPAction: "http://www.foo.com/ server/ errorNotification"

<s:Envelope
 xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:se="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:n="http://www.foo.com/header"
 xmlns:ns="http://www.foo.com/server"
 s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">


 <s:Header>
   <n:Request ID> 12345 </n:Request ID>
 </s:Header>
  <s:Body>
   <ns:Trouble Notification>
      <ErrorID> 1111 </ErrorID>
      <Description>Hard Disk Drive Trouble </Description>
   </ns:Trouble Notification>
 </s:Body>
</s:Envelope>
```

[Fig. 14]

EXEMPLARY PART DESCRIBING
RESPONSE TO CLIENT COMMAND

```
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
X-SOAP-Type : Response


<s:Envelope
 xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:se="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:n="http://www.foo.com/header"
 xmlns:ns="http://www.foo.com/server"
 s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">


 <s:Header>
   <n:Request ID> 12345 </n:Request ID>
 </s:Header>
  <s:Body>
   <ns:Trouble Notification Response>
      <Reception Result> OK </Reception Result>
   </ns:Trouble Notification Response>
 </s:Body>
</s:Envelope>
```

[Fig. 15]

EXEMPLARY PART
DESCRIBING SERVER COMMAND

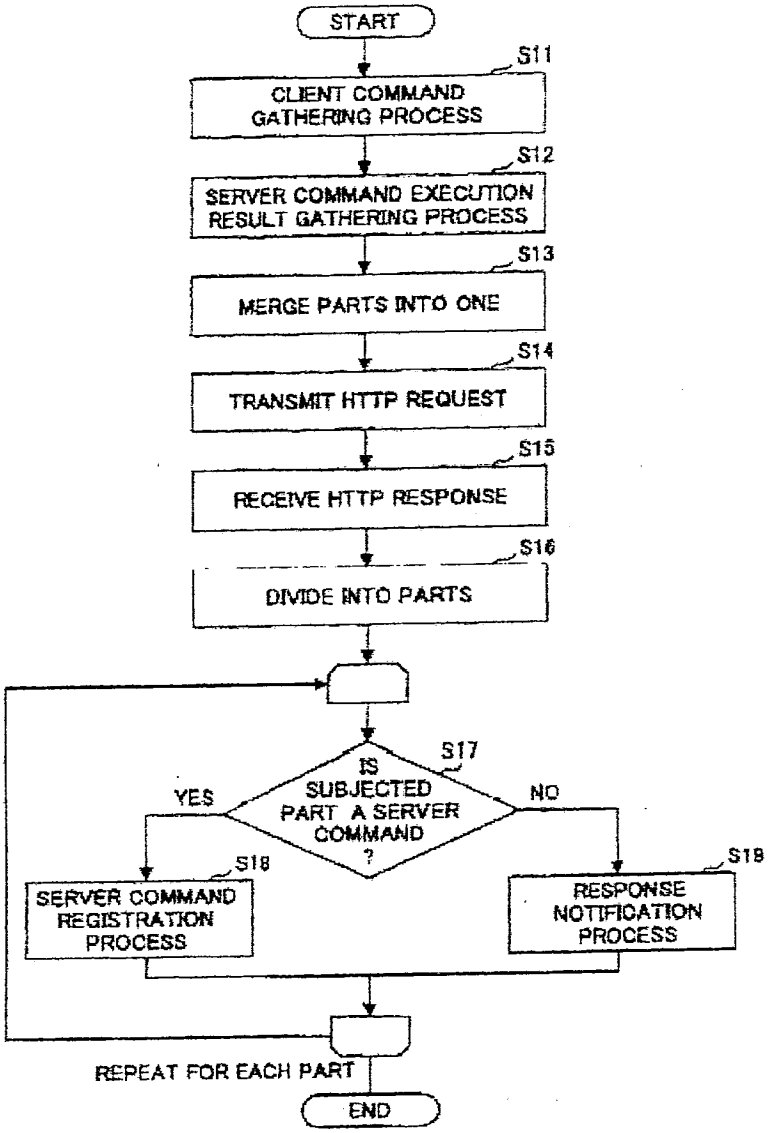
```
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
X-SOAP-Type : Request
SOAPAction: "http://www.foo.com/ client/getTemperature"

<s:Envelope
 xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:se="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:n="http://www.foo.com/header"
 xmlns:nc="http://www.foo.com/client"
 s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">


 <s:Header>
   <n:Request ID> 98765 </n:Request ID>
 </s:Header>
  <s:Body>
   <nc:Temperature Sensor Value Acqisition>
      <SensorID> 3 </SensorID>
   </nc:Temperature Sensor>
 </s:Body>
</s:Envelope>
```

[Fig. 16]

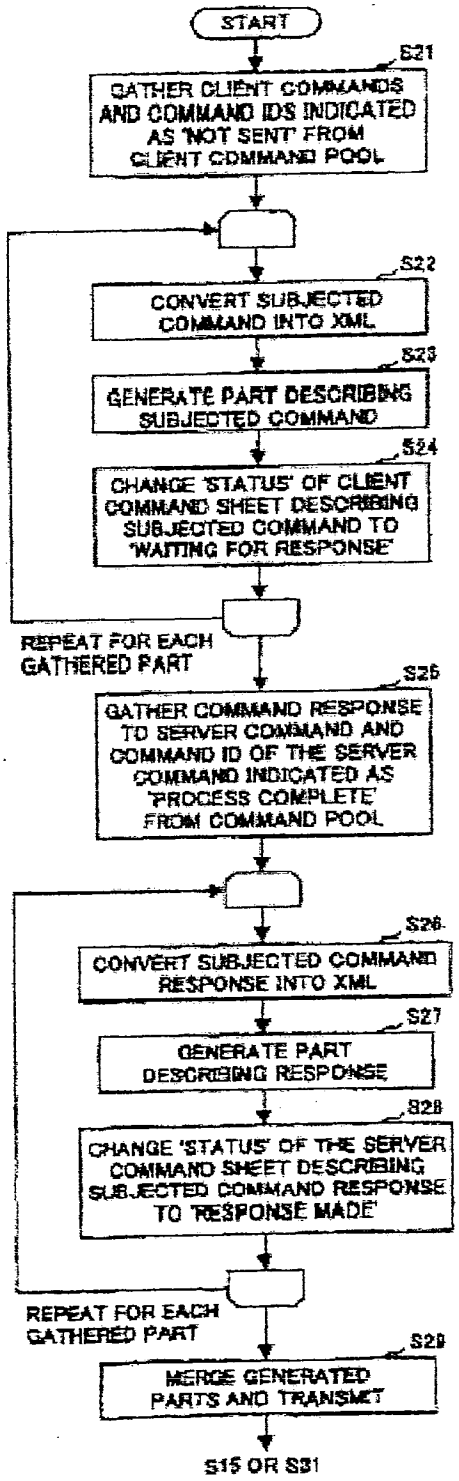
EXEMPLARY PART DESCRIBING
RESPONSE TO SERVER COMMAND

```
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
X-SOAP-Type : Response


<s:Envelope
 xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:se="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:n="http://www.foo.com/header"
 xmlns:nc="http://www.foo.com/client"
 s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">


 <s:Header>
   <n:Request ID> 98765 </n:Request ID>
 </s:Header>
  <s:Body>
   <nc:Temperature Sensor Value Acqisition Response>
      <Temperature> 52 </Temperature>
   </nc:Temperature Sensor Value Acqisition Response>
 </s:Body>
</s:Envelope>
```

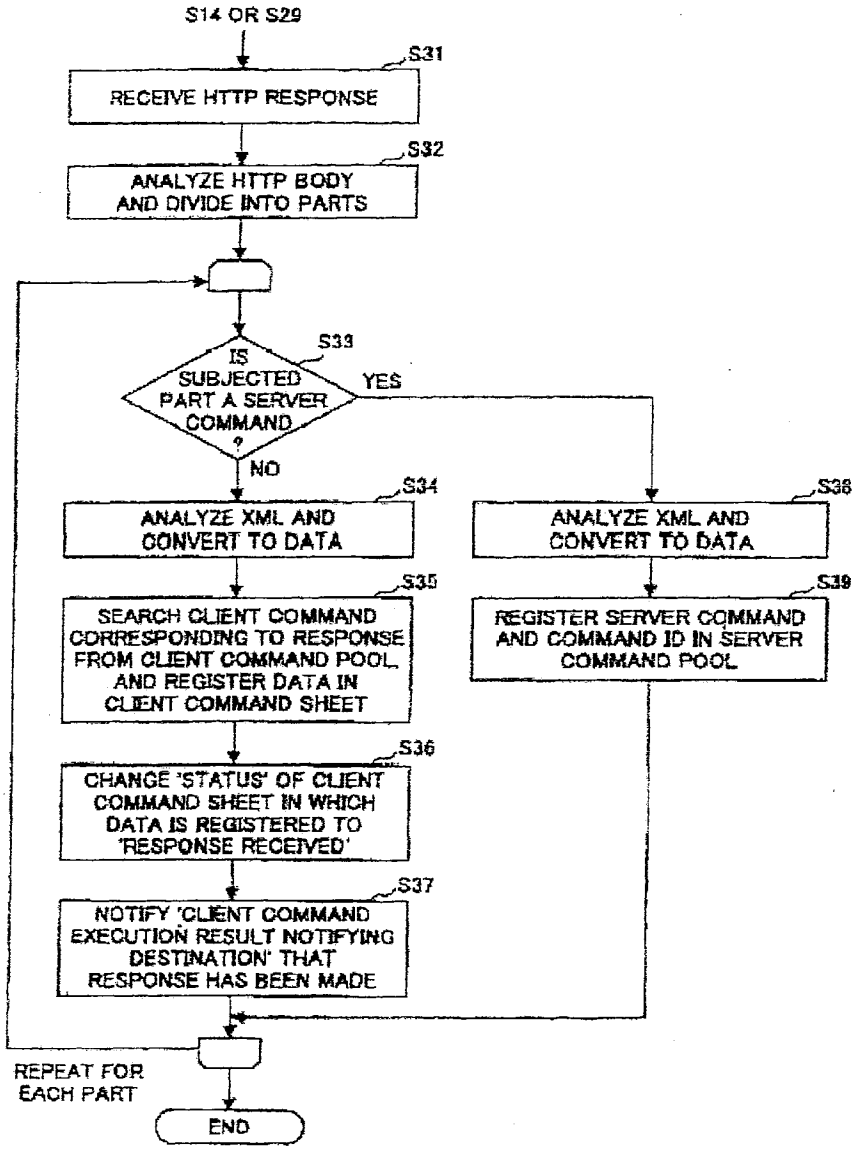
[Fig. 17]


START
S11
CLIENT COMMAND GATHERING PROCESS
S12
SERVER COMMAND EXECUTION RESULT GATHERING PROCESS
S13
MERGE PARTS INTO ONE
S14
TRANSMIT HTTP REQUEST
S15
RECEIVE HTTP RESPONSE
S16
DIVIDE INTO PARTS
S17
IS SUBJECTED PART A SERVER COMMAND ?
YES
NO
S18
SERVER COMMAND REGISTRATION PROCESS
S19
RESPONSE NOTIFICATION PROCESS
REPEAT FOR EACH PART
END

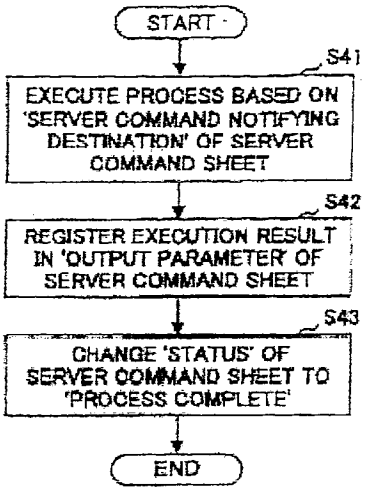
[Fig. 18]


START
S21
GATHER CLIENT COMMANDS AND COMMAND IDS INDICATED AS 'NOT SENT' FROM CLIENT COMMAND POOL
S22
CONVERT SUBJECTED COMMAND INTO XML
S23
GENERATE PART DESCRIBING SUBJECTED COMMAND
S24
CHANGE 'STATUS' OF CLIENT COMMAND SHEET DESCRIBING SUBJECTED COMMAND TO 'WAITING FOR RESPONSE'
REPEAT FOR EACH GATHERED PART
S25
GATHER COMMAND RESPONSE TO SERVER COMMAND AND COMMAND ID OF THE SERVER COMMAND INDICATED AS 'PROCESS COMPLETE' FROM COMMAND POOL
S26
CONVERT SUBJECTED COMMAND RESPONSE INTO XML
S27
GENERATE PART DESCRIBING RESPONSE
S28
CHANGE 'STATUS' OF THE SERVER COMMAND SHEET DESCRIBING SUBJECTED COMMAND RESPONSE TO 'RESPONSE MADE'
REPEAT FOR EACH GATHERED PART
S29
MERGE GENERATED PARTS AND TRANSMIT
S15 OR S31

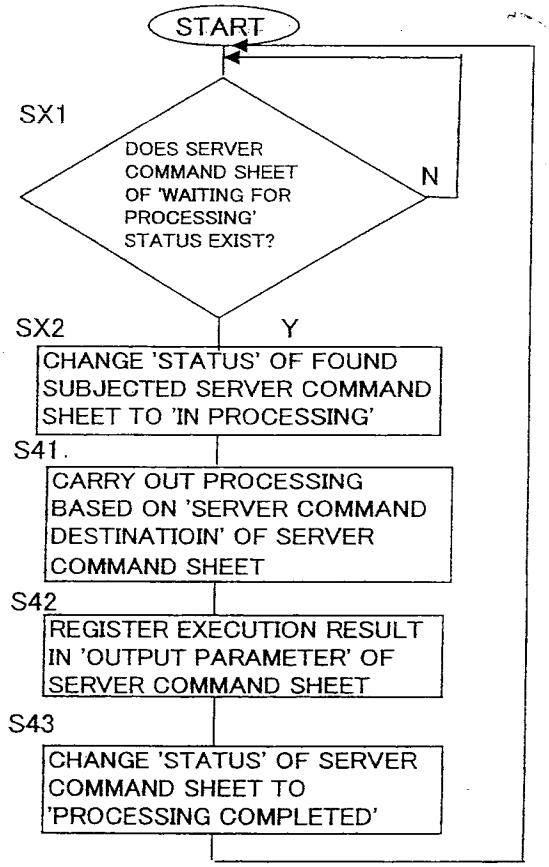
[Fig. 19]


S14 OR S29
S31
RECEIVE HTTP RESPONSE
S32
ANALYZE HTTP BODY AND DIVIDE INTO PARTS
S33
IS SUBJECTED PART A SERVER COMMAND ?
YES
NO
S34
ANALYZE XML AND CONVERT TO DATA
S35
SEARCH CLIENT COMMAND CORRESPONDING TO RESPONSE FROM CLIENT COMMAND POOL AND REGISTER DATA IN CLIENT COMMAND SHEET
S36
CHANGE 'STATUS' OF CLIENT COMMAND SHEET IN WHICH DATA IS REGISTERED TO 'RESPONSE RECEIVED'
S37
NOTIFY 'CLIENT COMMAND EXECUTION RESULT NOTIFYING DESTINATION' THAT RESPONSE HAS BEEN MADE
S38
ANALYZE XML AND CONVERT TO DATA
S39
REGISTER SERVER COMMAND AND COMMAND ID IN SERVER COMMAND POOL
REPEAT FOR EACH PART
END

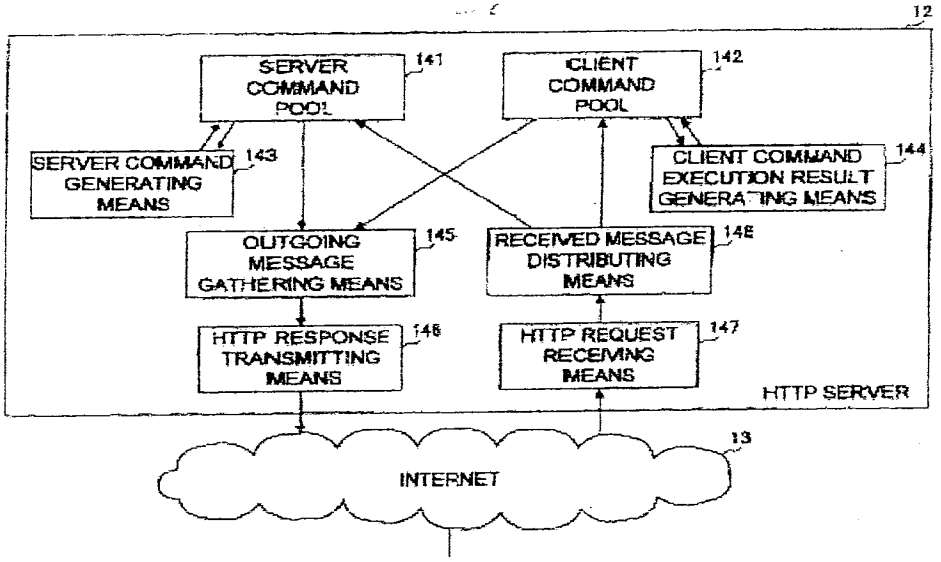
[Fig. 20]


START
S41
EXECUTE PROCESS BASED ON 'SERVER COMMAND NOTIFYING DESTINATION' OF SERVER COMMAND SHEET
S42
REGISTER EXECUTION RESULT IN 'OUTPUT PARAMETER' OF SERVER COMMAND SHEET
S43
CHANGE 'STATUS' OF SERVER COMMAND SHEET TO 'PROCESS COMPLETE'
END

[Fig. 21]


START
SX1
DOES SERVER COMMAND SHEET OF 'WAITING FOR PROCESSING' STATUS EXIST?
N
SX2
Y
CHANGE 'STATUS' OF FOUND SUBJECTED SERVER COMMAND SHEET TO 'IN PROCESSING'
S41
CARRY OUT PROCESSING BASED ON 'SERVER COMMAND DESTINATIOIN' OF SERVER COMMAND SHEET
S42
REGISTER EXECUTION RESULT IN 'OUTPUT PARAMETER' OF SERVER COMMAND SHEET
S43
CHANGE 'STATUS' OF SERVER COMMAND SHEET TO 'PROCESSING COMPLETED'

[Fig. 22]



[Fig. 23]

SERVER COMMAND SHEET IN HTTP SERVER

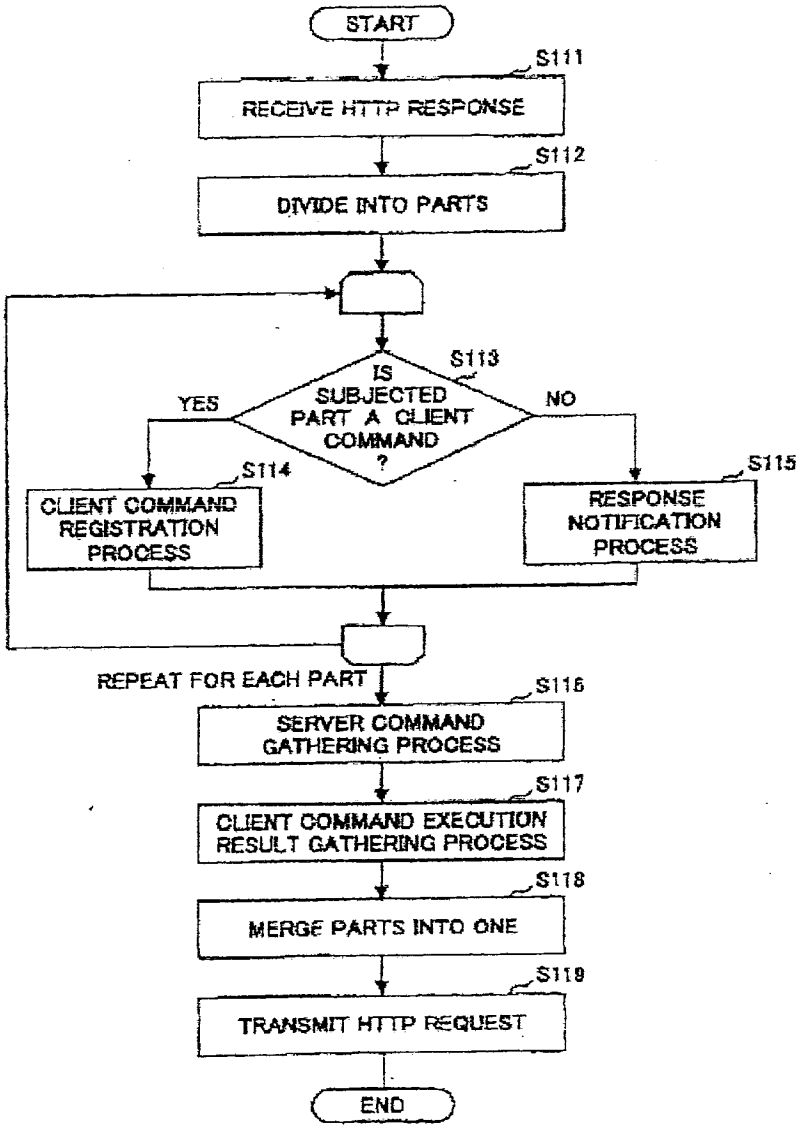
COMMAND ID
METHOD NAME (e.g., TROUBLE NOTIFICATION)
INPUT PARAMETER (e.g., TROUBLE CONTENT)
STATUS (INITIAL VALUE: NOT PROCESSED)
SERVER COMMAND EXECUTION RESULT NOTIFYING DESTINATION
OUTPUT PARAMETER (BLANK UNTIL RESPONSE IS ACQUIRED)

[Fig. 24]

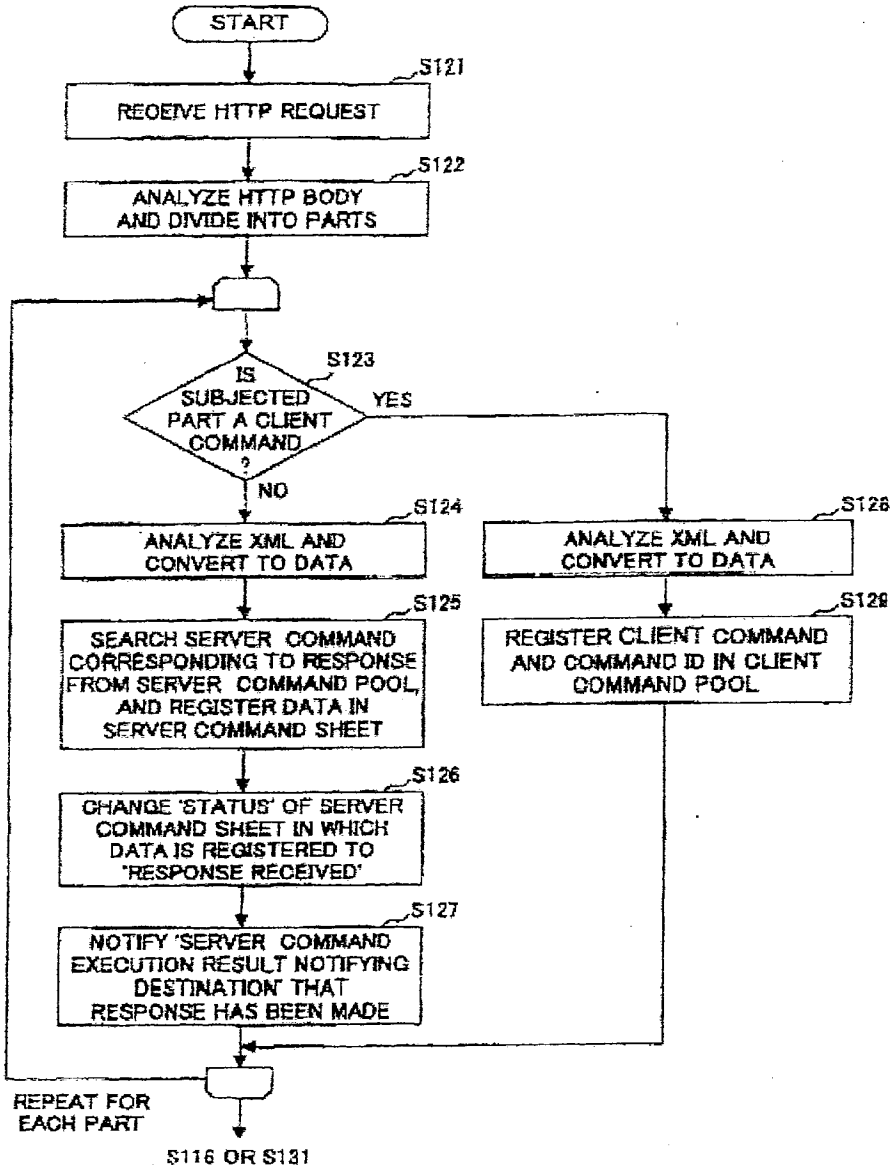
CLIENT COMMAND SHEET IN HTTP SERVER

COMMAND ID
METHOD NAME (e.g., COUNTER ACQUISITION)
INPUT PARAMETER
STATUS (INITIAL VALUE: NOT PROCESSED)
OUTPUT PARAMETER (BLANK UNTIL PROCESS IS COMPLETED)
CLIENT COMMAND NOTIFYING DESTINATION

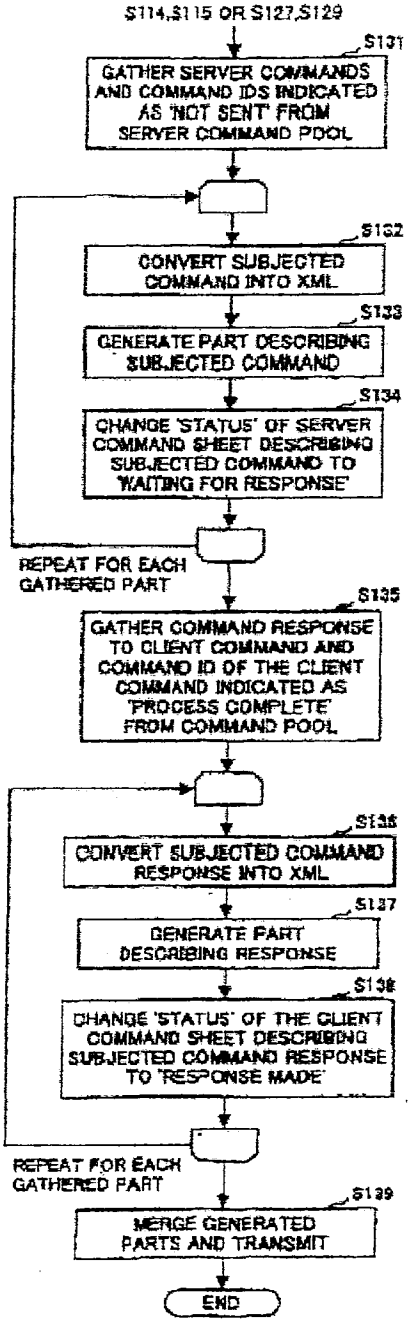
[Fig. 25]


START
S111
RECEIVE HTTP RESPONSE
S112
DIVIDE INTO PARTS
S113
IS SUBJECTED PART A CLIENT COMMAND ?
YES
NO
S114
CLIENT COMMAND REGISTRATION PROCESS
S115
RESPONSE NOTIFICATION PROCESS
REPEAT FOR EACH PART
S116
SERVER COMMAND GATHERING PROCESS
S117
CLIENT COMMAND EXECUTION RESULT GATHERING PROCESS
S118
MERGE PARTS INTO ONE
S119
TRANSMIT HTTP REQUEST
END

[Fig. 26]


START
S121
REOEIVE HTTP REQUEST
S122
ANALYZE HTTP BODY AND DIVIDE INTO PARTS
S123
IS SUBJECTED PART A CLIENT COMMAND ?
YES
NO
S124
ANALYZE XML AND CONVERT TO DATA
S125
SEARCH SERVER COMMAND CORRESPONDING TO RESPONSE FROM SERVER COMMAND POOL, AND REGISTER DATA IN SERVER COMMAND SHEET
S126
CHANGE 'STATUS' OF SERVER COMMAND SHEET IN WHICH DATA IS REGISTERED TO 'RESPONSE RECEIVED'
S127
NOTIFY 'SERVER COMMAND EXECUTION RESULT NOTIFYING DESTINATION' THAT RESPONSE HAS BEEN MADE
S128
ANALYZE XML AND CONVERT TO DATA
S129
REGISTER CLIENT COMMAND AND COMMAND ID IN CLIENT COMMAND POOL
REPEAT FOR EACH PART
S116 OR S131

[Fig. 27]


S114,S115 OR S127,S129
S131
GATHER SERVER COMMANDS AND COMMAND IDS INDICATED AS 'NOT SENT' FROM SERVER COMMAND POOL
S132
CONVERT SUBJECTED COMMAND INTO XML
S133
GENERATE PART DESCRIBING SUBJECTED COMMAND
S134
CHANGE 'STATUS' OF SERVER COMMAND SHEET DESCRIBING SUBJECTED COMMAND TO 'WAITING FOR RESPONSE'
REPEAT FOR EACH GATHERED PART
S135
GATHER COMMAND RESPONSE TO CLIENT COMMAND AND COMMAND ID OF THE CLIENT COMMAND INDICATED AS 'PROCESS COMPLETE' FROM COMMAND POOL
S136
CONVERT SUBJECTED COMMAND RESPONSE INTO XML
S137
GENERATE PART DESCRIBING RESPONSE
S138
CHANGE 'STATUS' OF THE CLIENT COMMAND SHEET DESCRIBING SUBJECTED COMMAND RESPONSE TO 'RESPONSE MADE'
REPEAT FOR EACH GATHERED PART
S139
MERGE GENERATED PARTS AND TRANSMIT
END

[Fig. 28]

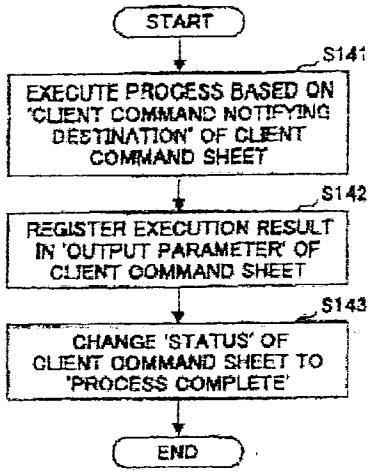
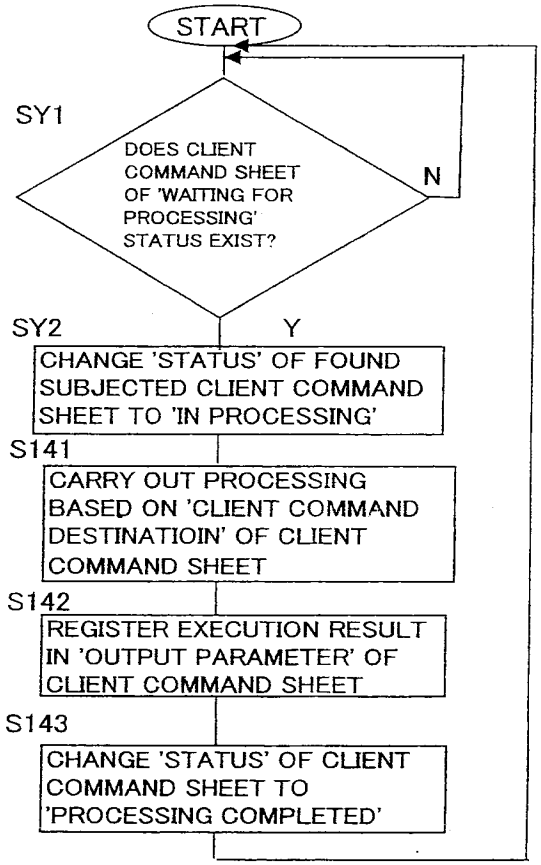
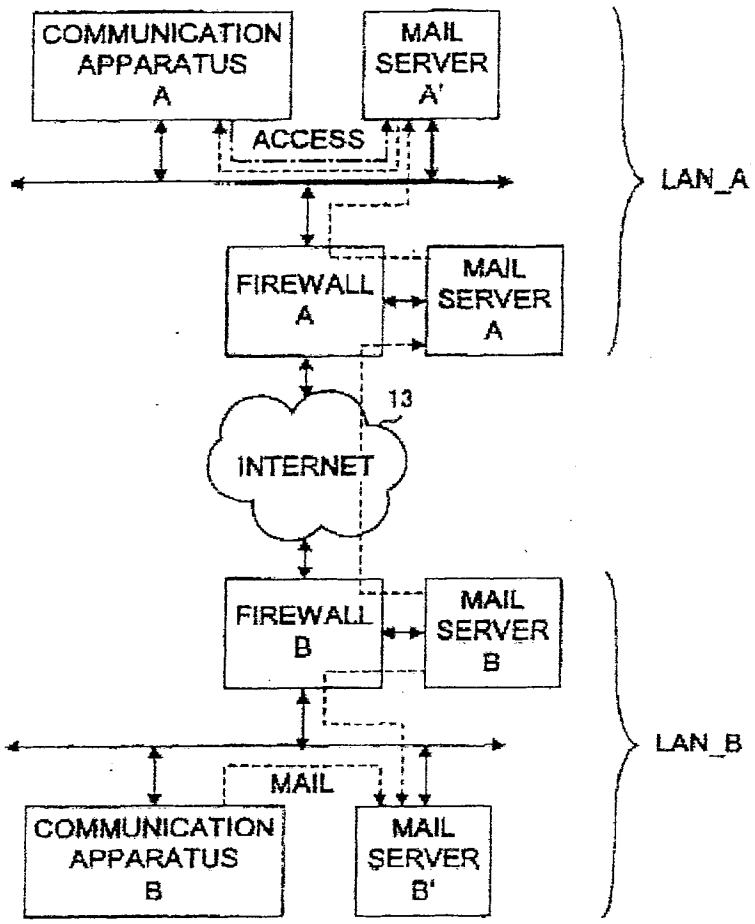

START
S141
EXECUTE PROCESS BASED ON 'CLIENT COMMAND NOTIFYING DESTINATION' OF CLIENT COMMAND SHEET
S142
REGISTER EXECUTION RESULT IN 'OUTPUT PARAMETER' OF CLIENT COMMAND SHEET
S143
CHANGE 'STATUS' OF CLIENT COMMAND SHEET TO 'PROCESS COMPLETE'
END

Fig. 29

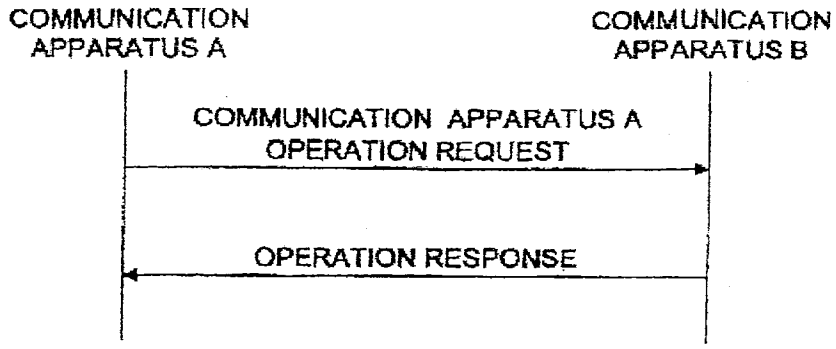

START
SY1
DOES CLIENT COMMAND SHEET OF 'WAITING FOR PROCESSING' STATUS EXIST?
N
SY2
Y
CHANGE 'STATUS' OF FOUND SUBJECTED CLIENT COMMAND SHEET TO 'IN PROCESSING'
S141
CARRY OUT PROCESSING BASED ON 'CLIENT COMMAND DESTINATIOIN' OF CLIENT COMMAND SHEET
S142
REGISTER EXECUTION RESULT IN 'OUTPUT PARAMETER' OF CLIENT COMMAND SHEET
S143
CHANGE 'STATUS' OF CLIENT COMMAND SHEET TO 'PROCESSING COMPLETED'

[Fig. 30]

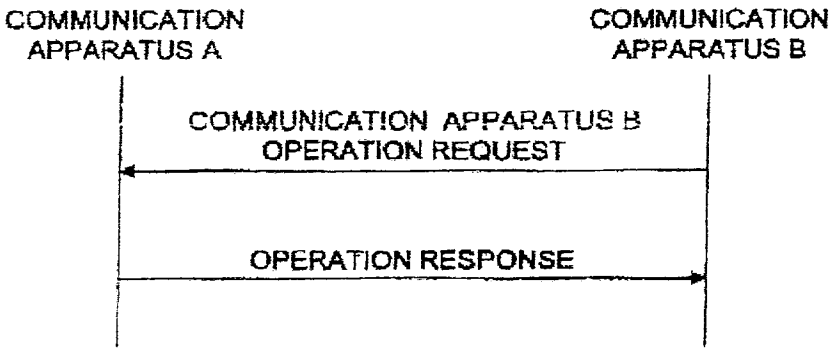


[Fig. 31]

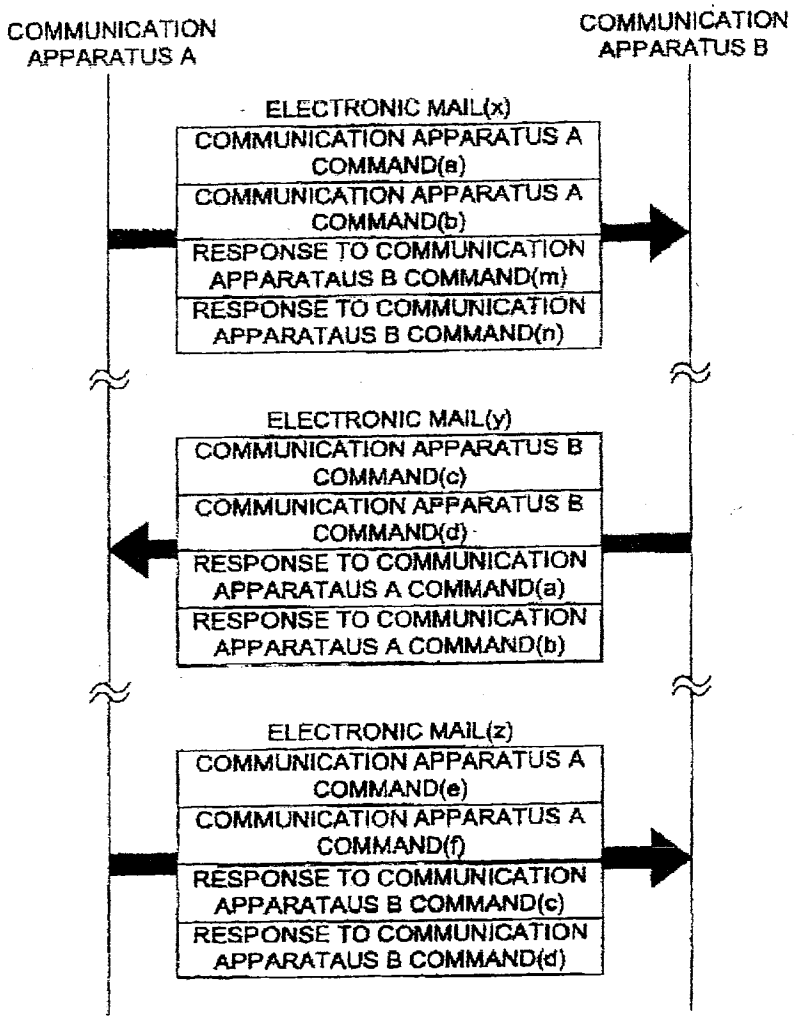
(A)



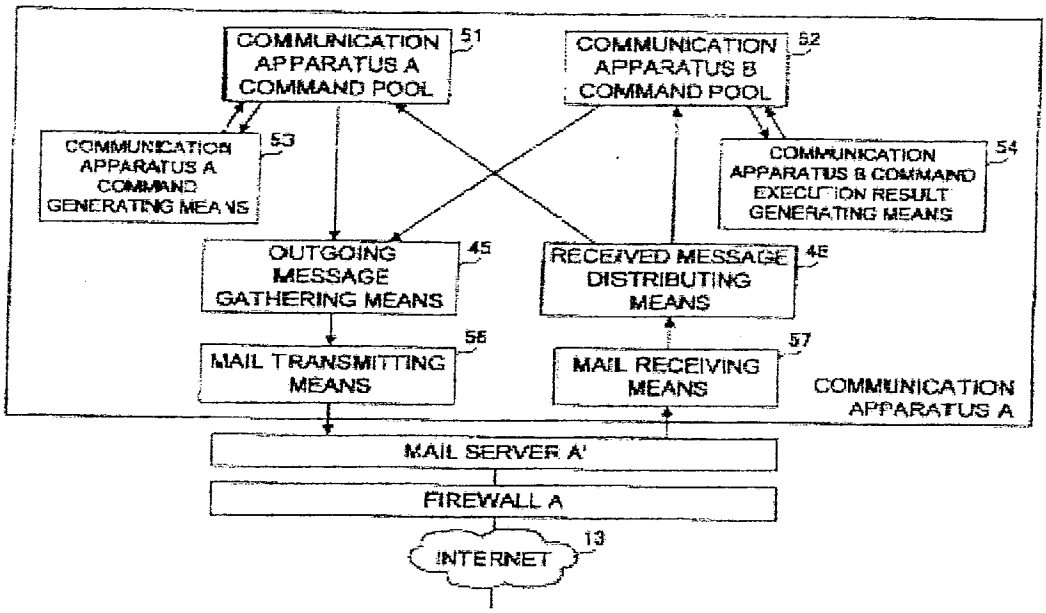
(B)


COMMUNICATION APPARATUS A
COMMUNICATION APPARATUS B
COMMUNICATION APPARATUS B OPERATION REQUEST
OPERATION RESPONSE

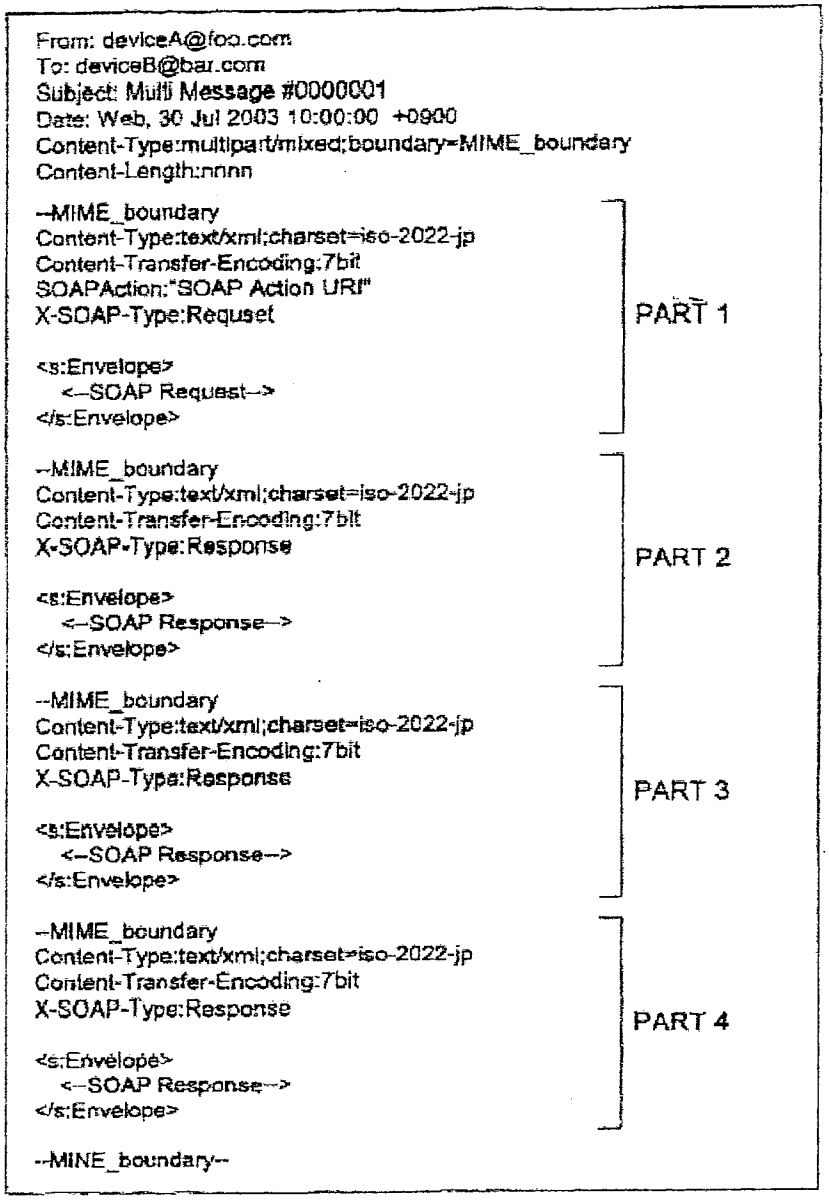
[Fig. 32]


COMMUNICATION APPARATUS A
COMMUNICATION APPARATUS B
ELECTRONIC MAIL(x)
COMMUNICATION APPARATUS A COMMAND(a)
COMMUNICATION APPARATUS A COMMAND(b)
RESPONSE TO COMMUNICATION APPARATAUS B COMMAND(m)
RESPONSE TO COMMUNICATION APPARATAUS B COMMAND(n)
ELECTRONIC MAIL(y)
COMMUNICATION APPARATUS B COMMAND(c)
COMMUNICATION APPARATUS B COMMAND(d)
RESPONSE TO COMMUNICATION APPARATAUS A COMMAND(a)
RESPONSE TO COMMUNICATION APPARATAUS A COMMAND(b)
ELECTRONIC MAIL(z)
COMMUNICATION APPARATUS A COMMAND(e)
COMMUNICATION APPARATUS A COMMAND(f)
RESPONSE TO COMMUNICATION APPARATAUS B COMMAND(c)
RESPONSE TO COMMUNICATION APPARATAUS B COMMAND(d)

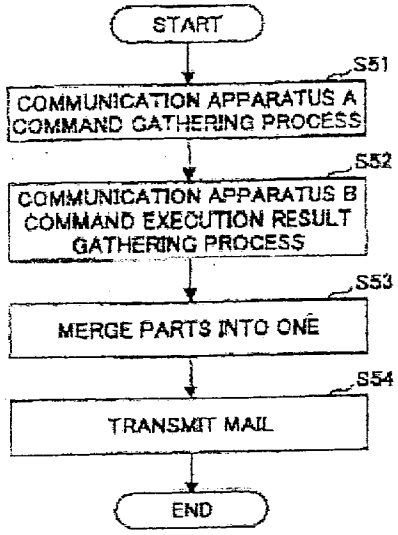
[Fig. 33]


COMMUNICATION APPARATUS A COMMAND POOL
51
COMMUNICATION APPARATUS B COMMAND POOL
52
COMMUNICATION APPARATUS A COMMAND GENERATING MEANS
53
COMMUNICATION APPARATUS B COMMAND EXECUTION RESULT GENERATING MEANS
54
OUTGOING MESSAGE GATHERING MEANS
45
RECEIVED MESSAGE DISTRIBUTING MEANS
46
MAIL TRANSMITTING MEANS
56
MAIL RECEIVING MEANS
57
COMMUNICATION APPARATUS A
MAIL SERVER A'
FIREWALL A
INTERNET
13

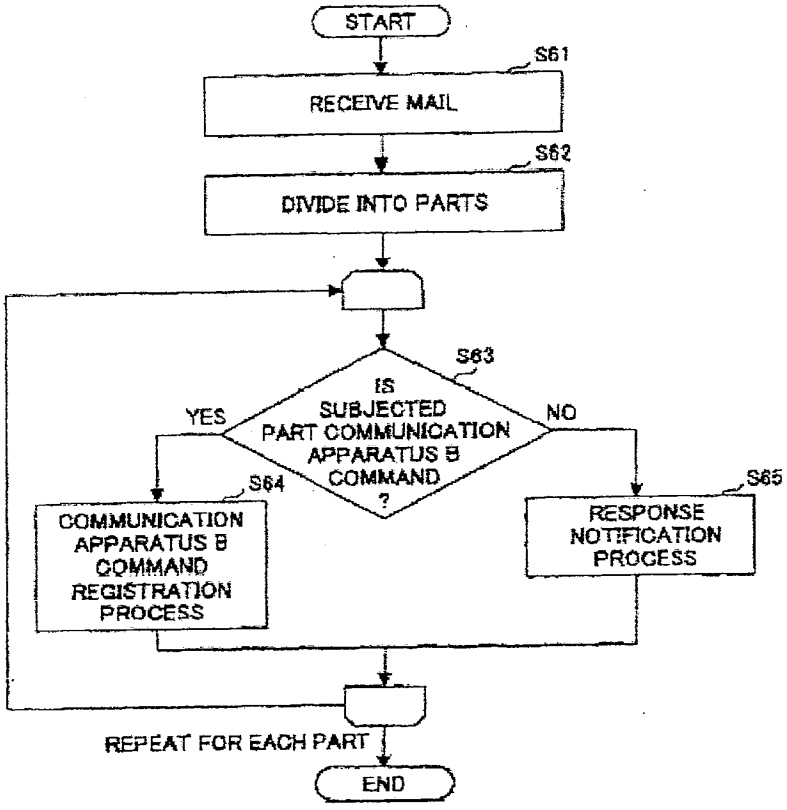
[Fig. 34]


From: deviceA@foo.com
To: deviceB@bar.com
Subject: Multi Message #0000001
Date: Web, 30 Jul 2003 10:00:00 +0900
Content-Type:multipart/mixed;boundary=MIME_boundary
Content-Length:nnnn
--MIME_boundary
Content-Type:text/xml;charset=iso-2022-jp
Content-Transfer-Encoding:7bit
SOAPAction:"SOAP Action URI"
X-SOAP-Type:Requset
<s:Envelope>
<--SOAP Request-->
</s:Envelope>
PART 1
--MIME_boundary
Content-Type:text/xml;charset=iso-2022-jp
Content-Transfer-Encoding:7bit
X-SOAP-Type:Response
<s:Envelope>
<--SOAP Response-->
</s:Envelope>
PART 2
--MIME_boundary
Content-Type:text/xml;charset=iso-2022-jp
Content-Transfer-Encoding:7bit
X-SOAP-Type:Response
<s:Envelope>
<--SOAP Response-->
</s:Envelope>
PART 3
--MIME_boundary
Content-Type:text/xml;charset=iso-2022-jp
Content-Transfer-Encoding:7bit
X-SOAP-Type:Response
<s:Envelope>
<--SOAP Response-->
</s:Envelope>
PART 4
--MINE_boundary--

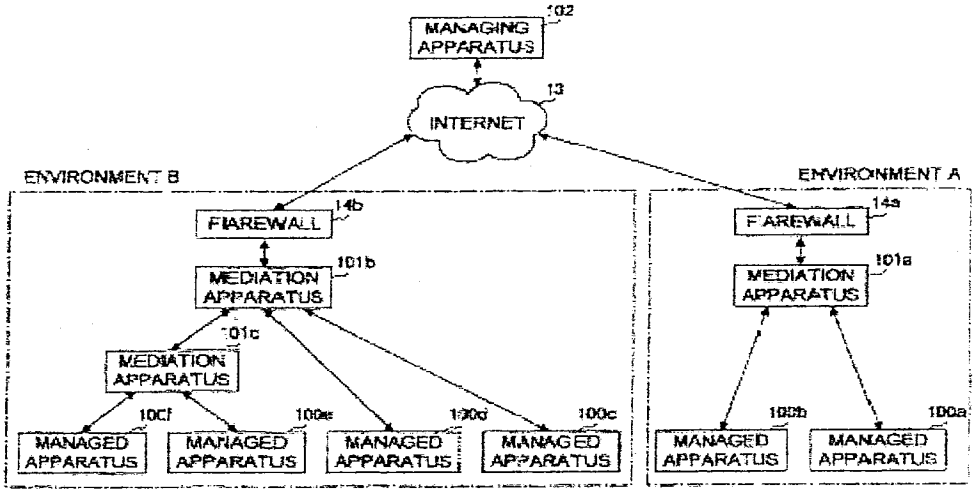
[Fig. 35]


START
S51
COMMUNICATION APPARATUS A COMMAND GATHERING PROCESS
S52
COMMUNICATION APPARATUS B COMMAND EXECUTION RESULT GATHERING PROCESS
S53
MERGE PARTS INTO ONE
S54
TRANSMIT MAIL
END

[Fig. 36]


START
S61
RECEIVE MAIL
S62
DIVIDE INTO PARTS
S63
IS SUBJECTED PART COMMUNICATION APPARATUS B COMMAND ?
YES
NO
S64
COMMUNICATION APPARATUS B COMMAND REGISTRATION PROCESS
S65
RESPONSE NOTIFICATION PROCESS
REPEAT FOR EACH PART
END

[Fig. 37]


MANAGING APPARATUS
102
INTERNET
13
ENVIRONMENT B
ENVIRONMENT A
FIAREWALL
14b
MEDIATION APPARATUS
101b
101c
MEDIATION APPARATUS
100f
100e
100d
100c
MANAGED APPARATUS
MANAGED APPARATUS
MANAGED APPARATUS
MANAGED APPARATUS
FIAREWALL
14a
MEDIATION APPARATUS
101a
100b
100a
MANAGED APPARATUS
MANAGED APPARATUS